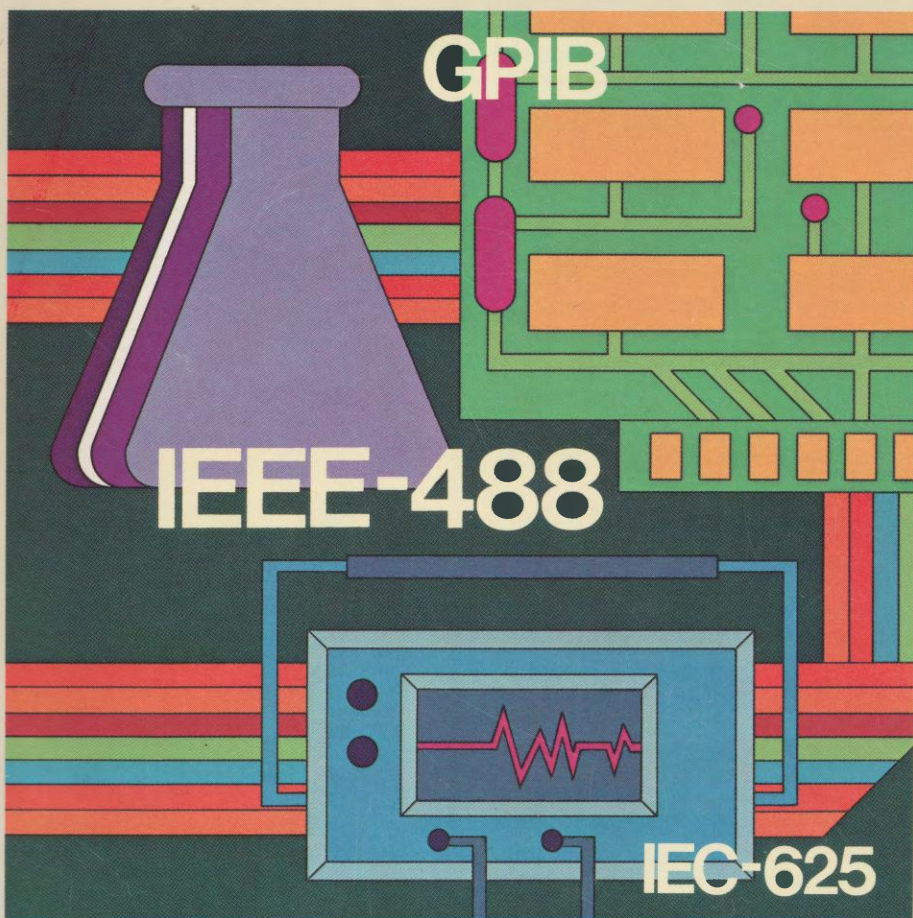


Apple II



# IEEE-488 Interface

User's Guide



## NOTICE

Apple Computer, Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

## DISCLAIMER OF ALL WARRANTIES AND LIABILITY

APPLE COMPUTER, INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL OR WITH RESPECT TO THE SOFTWARE DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. APPLE COMPUTER, INC. SOFTWARE IS SOLD OR LICENSED "AS IS". THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH THE BUYER. SHOULD THE PROGRAMS PROVE DEFECTIVE FOLLOWING THEIR PURCHASE, THE BUYER (AND NOT APPLE COMPUTER, INC. ITS DISTRIBUTOR, OR ITS RETAILER) ASSUMES THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL APPLE COMPUTER, INC. BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE, EVEN IF APPLE COMPUTER, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

This manual is copyrighted and contains proprietary information. All rights are reserved. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Apple Computer, Inc.

©1981 by Apple Computer, Inc.  
10260 Bandley Drive  
Cupertino, California 95014  
(408) 996-1010

The word APPLE and the Apple logo are registered trademarks of APPLE COMPUTER INC.

APPLE product #A2L0037  
(030-0197-A)

Apple II

---

# IEEE-488 Interface

---

User's Guide

PLEASE READ THIS MANUAL BEFORE ATTEMPTING TO INSTALL THE APPLE II IEEE-488 INTERFACE CARD IN THE APPLE COMPUTER. INCORRECT INSTALLATION COULD CAUSE PERMANENT DAMAGE TO BOTH THE INTERFACE CARD AND TO THE APPLE.

IEEE-488

# IEEE-488 INTERFACE CARD

## Specifications

The IEEE-488 Interface Card is a single-board computer interface card that provides a means of connecting a microcomputer to an IEEE-488 bus. The card is designed to be used with a variety of microcomputers, including the Apple II, Commodore 64, and IBM PC. It features a 16-bit data bus, a 16-bit address bus, and a 16-bit control bus. The card is capable of addressing up to 255 devices on the bus. It also includes a 16-bit timer/counter and a 16-bit shift register. The card is housed in a standard 8-inch by 8-inch format and is compatible with most microcomputer systems.

The IEEE-488 Interface Card is a single-board computer interface card that provides a means of connecting a microcomputer to an IEEE-488 bus. The card is designed to be used with a variety of microcomputers, including the Apple II, Commodore 64, and IBM PC. It features a 16-bit data bus, a 16-bit address bus, and a 16-bit control bus. The card is capable of addressing up to 255 devices on the bus. It also includes a 16-bit timer/counter and a 16-bit shift register. The card is housed in a standard 8-inch by 8-inch format and is compatible with most microcomputer systems.

The IEEE-488 Interface Card is a single-board computer interface card that provides a means of connecting a microcomputer to an IEEE-488 bus. The card is designed to be used with a variety of microcomputers, including the Apple II, Commodore 64, and IBM PC. It features a 16-bit data bus, a 16-bit address bus, and a 16-bit control bus. The card is capable of addressing up to 255 devices on the bus. It also includes a 16-bit timer/counter and a 16-bit shift register. The card is housed in a standard 8-inch by 8-inch format and is compatible with most microcomputer systems.

The IEEE-488 Interface Card is a single-board computer interface card that provides a means of connecting a microcomputer to an IEEE-488 bus. The card is designed to be used with a variety of microcomputers, including the Apple II, Commodore 64, and IBM PC. It features a 16-bit data bus, a 16-bit address bus, and a 16-bit control bus. The card is capable of addressing up to 255 devices on the bus. It also includes a 16-bit timer/counter and a 16-bit shift register. The card is housed in a standard 8-inch by 8-inch format and is compatible with most microcomputer systems.

The IEEE-488 Interface Card is a single-board computer interface card that provides a means of connecting a microcomputer to an IEEE-488 bus. The card is designed to be used with a variety of microcomputers, including the Apple II, Commodore 64, and IBM PC. It features a 16-bit data bus, a 16-bit address bus, and a 16-bit control bus. The card is capable of addressing up to 255 devices on the bus. It also includes a 16-bit timer/counter and a 16-bit shift register. The card is housed in a standard 8-inch by 8-inch format and is compatible with most microcomputer systems.

IF YOU ARE INTERESTED IN PURCHASING THIS CARD, PLEASE CONTACT THE MANUFACTURER AT THE FOLLOWING ADDRESS: IEEE-488 INTERFACE CARD, 12345 MAIN STREET, SUITE 100, DALLAS, TEXAS 75201. TEL: (214) 555-1234. FAX: (214) 555-5678.



# TABLE OF CONTENTS

## CHAPTER 1

### GETTING STARTED

---

1

- 1 What Is the Apple II GPIB?
- 2 Words of Advice
- 2 Special Symbols
- 3 What You Will Need
- 3 Installing the Interface Card

## CHAPTER 2

### THE IEEE-488 INTERFACE CARD

---

9

- 9 Controllers, Talkers and Listeners
- 10 Physical Requirements
- 12 Data Rates
- 13 Electrical Requirements
- 13 GPIB Connections
- 14 Data Lines
- 14 Management Control Lines
- 15 Transfer Control Lines
- 16 GPIB Functions

## CHAPTER 3

### PROGRAMMING THE GPIB

---

19

- 19 Getting Started
- 20 Command Notation
- 24 Addresses
  - 24 Listener Addresses
  - 24 Talker Addresses
  - 24 Secondary Addresses
- 25 On Using Addresses
- 25 GPIB Command Descriptions
  - 25 Sending A Command
  - 26 Command Protocol
  - 27 Bus Transfers

27 GPIB Commands  
 27 LINEFEED: LF  
 28 EOS: ES  
 28 SCREEN: SC  
 29 DEVICE: DV  
 29 WRITE: WT  
 30 WRITECNT: WC  
 30 READ: RD  
 31 READCNT: RC  
 32 XFER: XF  
 32 TRIGR: TG  
 32 CLRAL: CA  
 33 CLEAR: CL  
 33 REMAL: RA  
 33 REMDV: RM  
 34 LLKAL: LL  
 34 LOCAL: LA  
 34 LOCDV: LO  
 35 SRQD: SR  
 35 SPOLL: SP  
 36 PPOLL: PP  
 36 PPENB: PE  
 37 PPDIS: PD  
 37 PPUAL: PU  
 37 ABORT: AB  
 37 UNTALK: UT  
 38 GPIB Error Messages  
 38 The ONERR...GOTO Statement

## APPENDIX A

# ASCII CONVERSION TABLES

39

39 Table A-1 -- GPIB Listener Addresses  
 40 Table A-2 -- GPIB Talker Addresses  
 41 Table A-3 -- GPIB Secondary Addresses  
 42 Table A-4 -- ASCII Control Codes

## APPENDIX B

# BUS INPUT/OUTPUT SEQUENCES

43

## APPENDIX C

# 9914 REGISTERS

---

49

## APPENDIX D

# PROGRAMMING AIDS

---

51

- 51 Program 1:  
    CHR\$ Function For Integer BASIC
- 52 Program 2:  
    ONERR Routine
- 53 Program 3:  
    Skip Reinitialize Function
- 54 Program 4:  
    Input With Commas

## APPENDIX E

# FAST TRANSFER ROUTINES

---

55

- 55 GPIB Subr. Demo
- 56 Subroutines
- 56 General Control Values
- 57 Initialize 9914
- 58 Send Routine
- 59 Receive Routine
- 60 Wait Routine
- 61 Data Buffers
- 61 Subroutine Test
- 62 Symbol Table
- 62 Data Rates
- 63 Running and Testing

## APPENDIX F

# PROGRAMMING IN PASCAL

---

65

- 65 What You Must Do

## APPENDIX G

# SAMPLE PROGRAMS

67

- 67 Read Routine For HP 3438A
- 68 General Purpose HP 3586C Routine

## APPENDIX H

# SCHEMATIC DIAGRAM

71

# GLOSSARY

73

# INDEX

75

# CHAPTER 1

## GETTING STARTED

The Apple II IEEE-488 Interface Card provides a standard, plug-compatible, input/output bus for microcomputers. It allows your Apple II to control or communicate with up to 14 external devices. The external devices may be measuring instruments, data logging devices, or one or more of many available peripheral devices. These devices must all have bus interfaces which comply with the technical, electrical and mechanical requirements of IEEE Standard 488. Approximately 600 devices are currently available which use some form of this proposed standard input/output bus.

## WHAT IS THE APPLE II GPIB?

---

The formal name for this type of bus is "IEEE Standard Digital Interface for Programmable Instrumentation, 1978". The complete document, IEEE Standard 488-1978, may be obtained from the Institute of Electrical and Electronic Engineers, Inc. The instrument bus is also known as the "IEEE-488 Interface Bus", the "General Purpose Interface Bus" (GPIB) or the "Hewlett-Packard Interface Bus" (HP-IB®). You will not need to use the IEEE Standard 488 documentation unless you need actual design information for the construction of an interface device.

The Apple II IEEE-488 Interface will be referred to as the Apple II GPIB, or just the GPIB, throughout the rest of this manual. The Apple II GPIB consists of the interface circuit card and the interface cable. When we discuss the GPIB, remember that the terms "Apple II IEEE-488 Interface Card", "IEEE-488 bus" and "HP-IB®" all refer to the same type of universal I/O bus interface and bus operation. The interface circuit card itself will be referred to as the Apple II GPIB interface card, or just as the IEEE-488 card.

The Apple II GPIB interface card is the heart of the input/output operation. The interface card plugs into an Apple II connector slot and controls all transfers over the bus. The interface card is slot-independent; in other words, the interface card may be installed in any connector slot in the Apple (except slot 0). The interface card provides remote device control through simple BASIC commands from your program.

The interface card circuits perform the function of translating Integer BASIC or Applesoft BASIC instructions into commands understood by devices. The interface circuits include a controller which does all of the timing, control, and formatting for the instrument bus. The interface card also contains a read-only memory (ROM) that provides the control information needed by the interface controller circuit to interpret BASIC commands.



## WORDS OF ADVICE

Although the electrical characteristics of the bus are clearly specified by IEEE Standard 488, the addressing conventions, input data format, and output data format can vary from one device to the next. You must carefully study the manufacturer's operating manual for each device before attempting to place the device in service on the bus. For example, the output data may be ASCII or binary. If it is binary, the least significant bit of each byte may be transmitted first or last, and so on.

There are no really good reference books available for setting up and operating a system under the IEEE-488 proposed standard. You must have a thorough command of BASIC or APPLESOFT programming and a working knowledge of DOS. If you do not, then refer to the BASIC and DOS manuals and practice the examples. You may also need to understand the relationship between ASCII, hexadecimal, octal, and binary notation. Conversion tables are included in an appendix.

A test device which can be plugged into the bus and which can be used to test the commands and data sent by your program may be advantageous. A device that can single-step the bus program is almost a necessity for complex systems. One such test device now available is the model ZT488 GPIB ANALYZER manufactured by Ziatech Corporation. Operation of the bus may also be tested by use of a Tektronix or Hewlett-Packard logic analyzer. Another useful device is the Gould Model K100D/408 GPIB Analyzer.

## SPECIAL SYMBOLS

This manual uses some special symbols to indicate particularly noteworthy pieces of information. If you see the symbol



it means that the following paragraph discusses some possibly unexpected Apple behavior. The symbol



means that the following paragraph contains special information that may be useful to you. Read these sections carefully.

# WHAT YOU WILL NEED

---

The Apple II IEEE-488 Interface Card is designed for experienced users who are familiar with APPLESOFT or Integer BASIC, and DOS 3.2 or DOS 3.3. Those who wish to use the card with the Apple Pascal Operating System should be familiar with Apple Pascal, its operating system, and 6502 assembly language.

The programs described in this manual, as well as the firmware used by the card, will run on an Apple II or an Apple II Plus with at least 32K memory, and at least one Disk II drive and controller.

Six items make up your Apple II IEEE-488 Interface Card package:

1. Apple II IEEE-488 Interface Card  
Part # 030-0670-0016
2. Interconnection cable with metal clamp on it  
Part # 030-0590-0030
3. Two IEEE standard screws
4. Two non-standard screws
5. A warranty card
6. This manual

Carefully unpack all the items in your shipment. Fill out the warranty card and mail it to your Apple dealer.

## INSTALLING THE INTERFACE CARD

---

Read this section carefully, even if you have installed Apple peripheral interface cards before. The Apple II GPIB interface card is easy to install, but it is important to install it correctly.



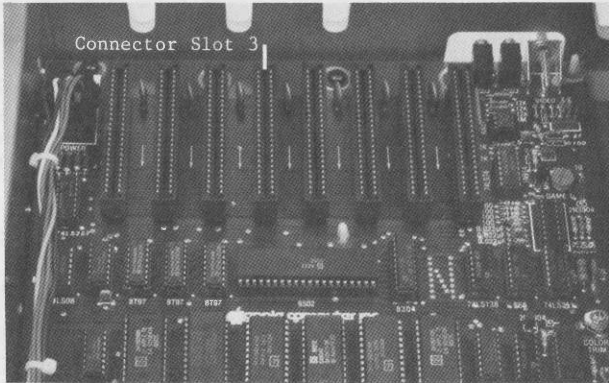
Before connecting or disconnecting anything on the Apple, turn off the power with the switch on the back left corner of the Apple case. **THIS IS NECESSARY.** If you try to connect or disconnect anything from the inside of your Apple when the power is on, you are likely to damage the circuits and lose anything currently stored in random access memory (RAM).

Do not unplug the Apple; just turn it off. If you unplug the Apple, you will isolate it from earth ground and leave it vulnerable to static discharges.

Remove the Apple cover by pulling up on the two back corners of the cover until the two corner fasteners pop apart. Slide the cover back until it is free of the case and lift the cover off.

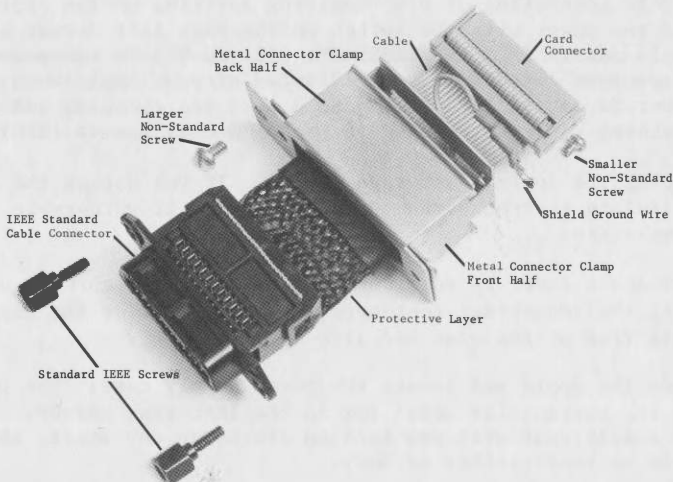
Look inside the Apple and locate the power supply case. The power supply is the rectangular metal box in the left rear corner. Touch the power supply case with one hand to discharge any static charge that may be on your clothes or body.

Along the back inside edge of the Apple you will see eight long narrow slots called "connector slots". The connector slots are numbered from 0 at the left corner to 7 at the right corner. The numbers are printed along the back edge behind the connector slots. The connector slots are shown in the following photograph.



Your disk controller card is plugged into slot 6. If you have more than two drives, a second disk controller card is plugged into slot 4 or 5. If you have an Apple Language Card, it is plugged into slot 0. If you have a printer, the printer interface card is probably plugged into slot 1. This manual assumes that you will plug the Apple II GPIB interface card into slot 3. If you are going to plug your card into another slot, substitute that slot's number for all references to slot 3 in this manual.

The Apple IEEE interconnection cable is shown below.

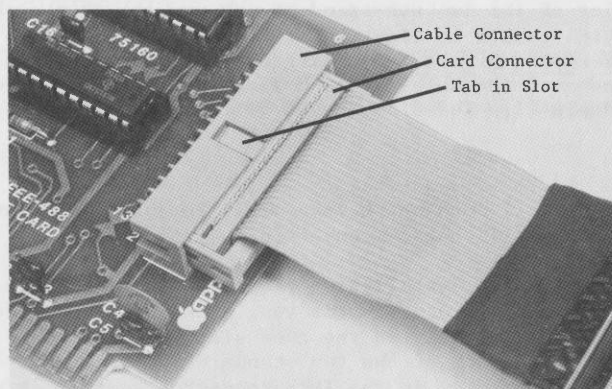


The IEEE standard cable connector is at the left. This connector is the one to which other devices will be attached. Just to the right of the IEEE connector is a two piece metal clamp. We will refer to the part of the clamp nearest the IEEE connector as the back half of the clamp. The other half of the connector clamp, the front half, has an narrow cable pathway with the cable going through it. At the right end of the cable is the interface card's connector.

When the two halves of the clamp are nestled snugly together, as they are in the photograph above, they form a rectangular enclosure with an opening for the IEEE connector on the left, an opening for the cable on the right, and two other large openings. These two openings allow the clamp to slide into one of the slots in the back of the case itself. Be sure that the two halves are together, but not too tightly; the case must fit into the openings.

The left end of the cable, by the IEEE connector, is covered by a protective layer, beneath which one of the wires, the shield ground, is severed. A new wire, attached to the IEEE connector end of the severed wire, extends from beneath the protective covering. Make sure that this wire runs parallel to the IEEE cable, and through the cable pathway in the front half of the metal connector clamp. This wire will be attached to case ground.

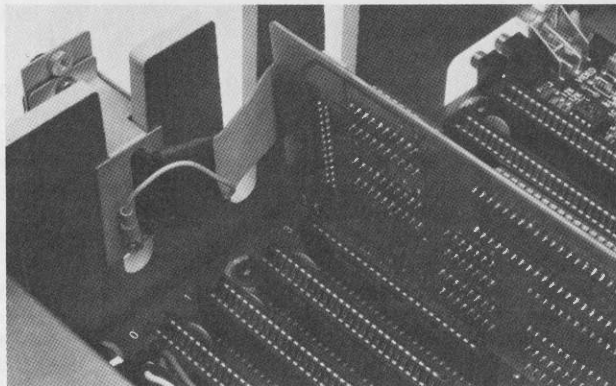
The card connector on the bus cable plugs into a connector on the end of the interface card. Hold the bus cable so that the tab on the cable's card connector is aligned with the slot on the card's connector. Press the cable's connector into the card's connector until it is firmly seated, as shown in the next photograph.



Handle the interface card as you would handle an expensive phonograph record. Grasp it only by the corners or edges, and do not touch the components or pins, especially the edge connector fingers.

Grasp the upper edge of the interface card in your right hand, and the metal connector clamp for the IEEE connector in your left. Slide the metal connector clamp into an available slot in the back of the Apple's case; if the clamp does not slide easily, pull the front and back halves a little farther apart, making the opening for the case larger. Simultaneously guide the card towards connector slot 3 with your right hand.

Insert the gold "fingers" of the interface card edge connector into slot 3, rear edge first. Gently push the front edge of the card down until it is level and firmly seated as shown in the photograph.



Using the smaller of the two non-standard screws, attach the shield ground wire, which should be right alongside the bus cable, to the hole in the lower part of the front half of the connector clamp. By doing so, you connect the shield ground line for the bus to the case ground of the Apple II. The line should be connected as shown in the photo above.

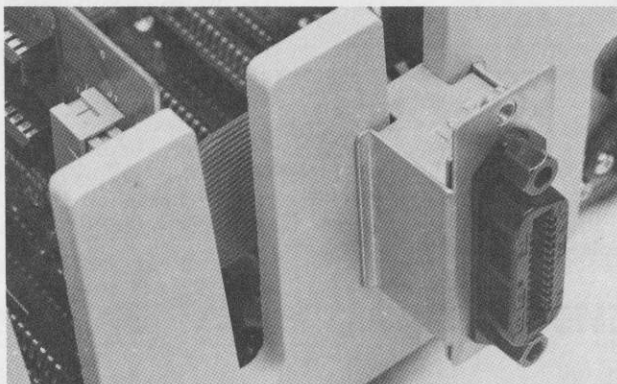


If your Apple II does not have a metallic interior, your shield ground will not be connected to ground.

For the final step of the installation process, turn the Apple around so the rear of the case points towards you. Make sure that the metal connector clamp rests securely in the case slot, positioned as far down as possible. Take one of the two standard IEEE screws, place it through one of the two holes in the IEEE connector so that it goes into the corresponding hole in the metal connector clamp, and tighten it. Repeat this procedure with the other IEEE screw and other connector hole.



You have just accomplished two things. You have tightened the metal connector clamp against the back of the case, preventing the connector from moving in the slot, and improving the connection between the clamp and case ground. You have also fastened the IEEE connector firmly in place so that there is no chance of putting undue strain on the cable. The connection, as seen from the rear of the Apple, appears in the next photograph.



The remaining screw may be used to suppress the radio frequency interference signals which radiate from the bus cable. If used, the screw should join the shield of the cable connected to the IEEE connector in the back of the Apple to the top hole in the front half of the metal connector clamp.

Slide the Apple case top plate in place and press down on the rear corners until the corner fasteners pop into place. The Apple II IEEE-488 Interface Bus is now installed and ready to use.



## CHAPTER 2

# THE IEEE-488 INTERFACE CARD

The Apple II IEEE-488 Interface Card is one of several "standardized" input/output (I/O) interfaces which have been developed since 1972 in the effort to provide a universal I/O bus.

The advantage of the Apple II GPIB is that it conforms to IEEE Standard 488 and that a single interface card lets your Apple II control and communicate with up to 14 other data gathering or processing units. It uses a commercially available standard interface cable with piggyback connectors which allow the devices on the bus to be daisy chained or star connected as shown in Figures 2-2 and 2-3. The standard cable connects to all devices designed to operate under the IEEE-488 proposed standard.

## CONTROLLERS, TALKERS AND LISTENERS

---

The Apple II GPIB uses eight data lines and eight control lines to transfer commands and data to and from the I/O devices on the bus. A block diagram of the bus is shown in Figure 2-1. The devices on the bus can be described in terms of their primary functions of controller, talker, and listener.

Any device on the bus is designated a "talker" when its primary function is the transmission of data. An example of a talker is a digital voltmeter. If a talker transmits data over the bus continuously, no other devices can use the bus; therefore, the talker function is turned on by the controller when necessary.

Any device on the bus is designated a "listener" when its primary function is the execution of commands. A printer is such a device. If a listener accepted all commands and data transmitted on the bus, chaos would result; therefore, the listener function must also be turned on by the controller. Some listeners also have a talker function.

The "controller" is the device that arbitrates all transfers of data on the bus. The controller is the device that can turn on a talker for the transmission of data. Likewise, the controller is the device that can turn on a listener for the reception of data.



Only one controller is allowed on this implementation of the bus. This controller must be your Apple.

A controller is able to interact with both talkers and listeners in various ways. The exact capabilities of the Apple as controller are detailed in the chapter on programming the GPIB.

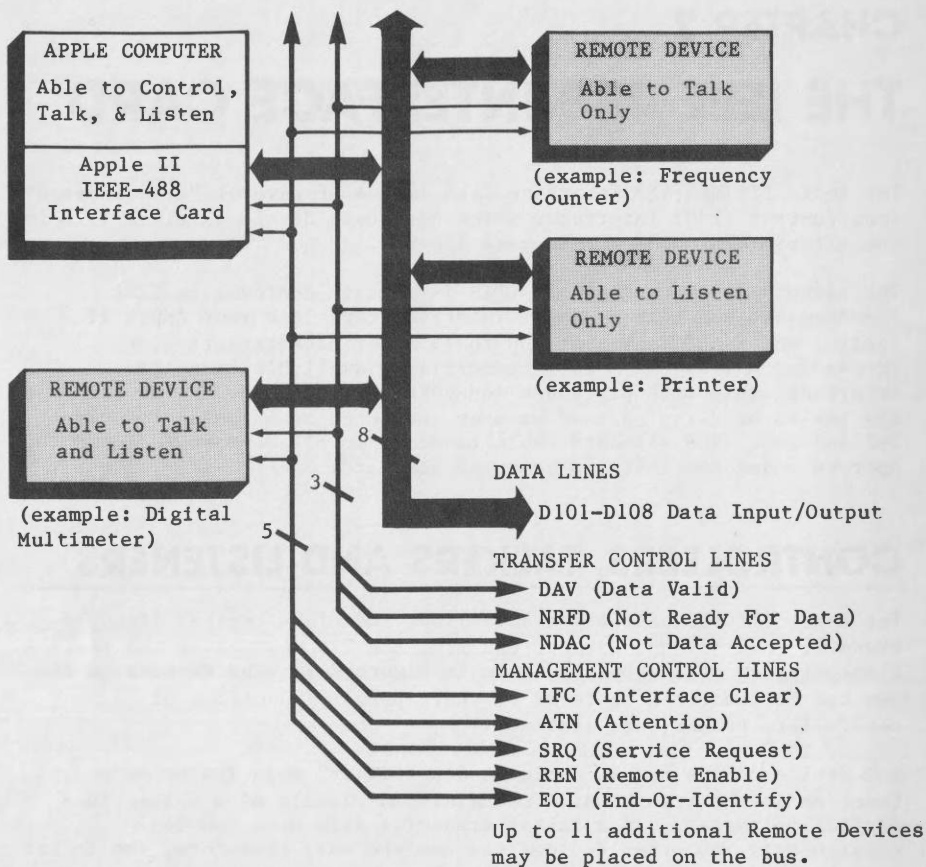


Figure 2-1. Apple II GPIB Functional Block Diagram.

## PHYSICAL REQUIREMENTS

We have already stated that the Apple II GPIB can control up to 14 external devices. There are some limits on the total length of the bus and the distance between devices. The standard states that the total length of the bus may not exceed two meters times the number of devices on the bus. The maximum length of the entire bus is 20 meters.

The devices on the bus can be connected in either a daisy chain or a star configuration using the stackable connectors of the available standard bus cables. Daisy chain connections are preferred by most system designers, since the star connector stack may be unwieldy in larger systems. A sketch of star and daisy chain connected equipment are shown in Figures 2-2 and 2-3.

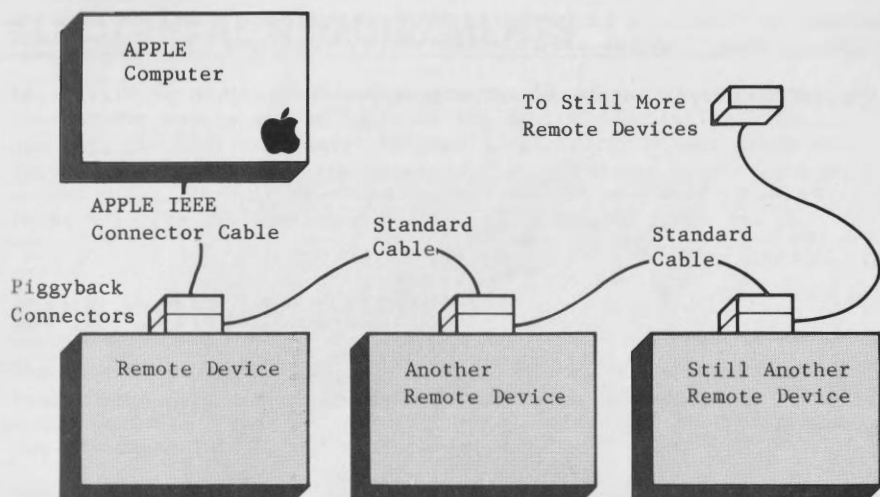


Figure 2-2. Daisy Chain Configuration.

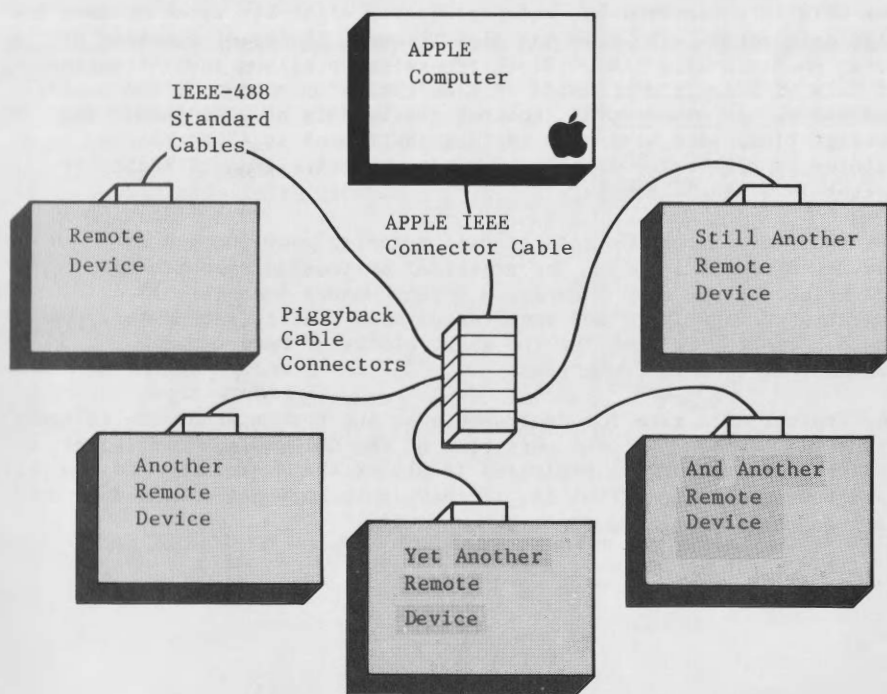
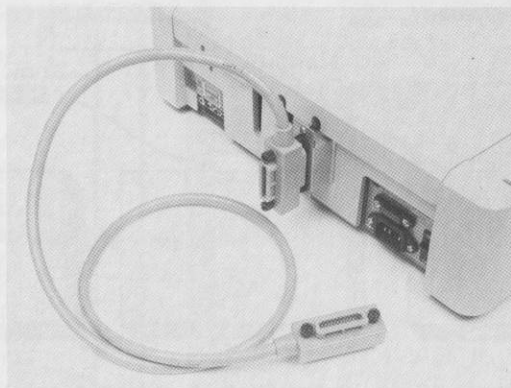


Figure 2-3. Star Connected Configuration.



Stackable or piggyback cables in various lengths are available from electronic supply houses and, in some cases, from your Apple dealer.

A typical commercial cable assembly is shown in the photograph.



## DATA RATES

The GPIB is a parallel bus which transmits eight-bit bytes of data at high data rates. The GPIB can also transmit blocks of commands or data, one byte at a time. Block transmission allows rapid transfer of data with a minimum number of time-consuming handshake and control functions. In other words, control overhead is at a minimum. The average block size sent by a typical instrument is 13 bytes. A printer or disk drive is able to send or receive an entire line or sector in a single block.

The theoretical maximum rate of data transfer over the bus is one megabyte per second. The practical maximum is approximately 250 kilobytes per second during a Direct Memory Access (DMA) operation. Usually, a DMA requires so much time to set up that the speed advantage is lost for the short blocks of data usually transferred in GPIB operations.

The typical data rate for instruments on the bus is about one to two kilobytes per second. The operation of the GPIB is most efficient, and program overhead is minimized if all of the devices on the bus have buffered inputs; that is, if they can accept a block of data as fast as the controlling Apple can send it.

# ELECTRICAL REQUIREMENTS

Ideally, devices on the bus that are not in use should be turned off to conserve energy and to increase the device's operating life. However, the best compromise between ideal energy conservation and proper operation of the bus requires that only some of the devices be turned off. The rule of thumb is that the Apple itself, plus at least half the devices connected to it, must have power on.

## GPIB CONNECTIONS

The interface bus consists of eight data lines and eight control lines. The organization of these lines on the standard bus cable is shown in Table 2-1. The mnemonic symbols given for the connections are explained in the following paragraphs.

<u>PIN</u>	<u>SIGNAL</u>
1	DIO1
2	DIO2
3	DIO3
4	DIO4
5	EOI
6	DAV
7	NRFD
8	NDAC
9	IFC
10	SRQ
11	ATN
12	shield (earth ground)
13	DIO5
14	DIO6
15	DIO7
16	DIO8
17	REN
18	logic ground
19	logic ground
20	logic ground
21	logic ground
22	logic ground
23	logic ground
24	logic ground

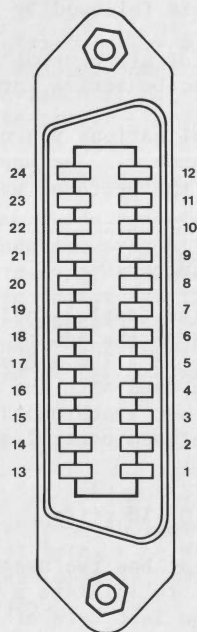


Table 2-1. Standard Bus Connector Pin Assignments.

## DATA LINES

There are eight data lines in the GPIB. The data lines are numbered DIO1 (Data Input/Output 1) through DIO8 and correspond to the data bits 0 through 7 used by the Apple. The data carried by the data lines is represented using active low or negative logic. In negative logic, a low (0 volt) level on a line corresponds to a True and a high (5 volt) level corresponds to a False. The interface card inverts the data before sending it to and after receiving it from the Apple; thus all the data the Apple sends and receives is positive true logic.

## MANAGEMENT CONTROL LINES

The GPIB uses eight lines to control the devices on the bus and to direct the flow of data on the bus. Five of the control lines are used for the management of data transfers. The remaining three control lines are used for handshaking, or coordination of the transfer of individual bytes of data. A brief description of the five management control lines is given below. The mnemonic name for each signal is followed by its meaning.

The signals of all control lines are active low; that is, a line is considered to be active, or True, when its value is low (0 volts).

For most applications you need not know which control lines are used with each command. The appendix on bus input/output sequences gives examples of the protocol used by each command.

### ATN      ATteNtion

The controller (Apple) indicates that an address or control byte is being placed on the bus by setting the ATN line true. The ATN line is set false to allow a device to place data on the bus. The ATN line is also used with the EOI line to initiate a parallel poll. The ATN line is set synchronously with the control handshake to eliminate confusion between control and data bytes.

### EOI      End Or Identify

The EOI signal has two uses. 1) The controller (Apple) sets EOI true and ATN true to initiate a parallel poll. 2) A talker will set EOI true with the last byte of data to indicate the end of the transmission. Unfortunately, not all available devices use the EOI to end a transmission.

### SRQ      Service ReQuest

The SRQ signal is set true by any device on the bus which requires some action from the controller. Upon reception of the SRQ signal, the controller software places the address of each device in turn (serial poll) on the data lines of the bus. Each device returns one status byte. When the device that set SRQ true detects its address on the bus, it identifies itself by returning a status byte with DIO7 true, and then it releases the SRQ line.

### IFC      InterFace Clear

The IFC signal is used by the controller (only) to set all device interfaces to a known initial state. This signal is used by the Abort command only.

### REN      Remote ENable

The REN signal is used by the controller (only) to allow a device to be remotely controlled. The device remains remotely controlled as long as the REN signal remains true. The device will not respond to data sent on the bus unless the REN signal is true.

## TRANSFER CONTROL LINES

Data transfer on the bus is controlled by the three transfer control lines. The three lines use a three-wire handshake system (patented by Hewlett-Packard). The handshake system is asynchronous, allowing devices with both fast and slow data rates to operate on the same bus. The three-wire handshake is used to transfer every byte of information that is passed over the data lines. The three transfer control lines are described below. A diagram of the handshaking protocol is shown in figure 2-4.

### NRFD      Not Ready For Data

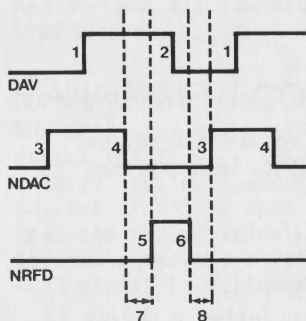
The NRFD signal is set true (low) by a listener to indicate that it is not yet ready for the next data or control byte. The NRFD signal at the controller will not be sensed as false (that is, ready for data) until all listeners on the bus are ready.

### NDAC      Not Data ACcepted

The NDAC signal is set true (low) by a listener to indicate that it has not yet accepted the data or control byte on the data bus. The NDAC signal at the controller will not be sensed as false (that is, data accepted) until all listeners on the bus have accepted the data.

## DAV      Data Valid

The DAV signal is set true (low) by a talker to indicate that a data or control byte is on the data bus and is available to all listeners.



1. Data No Longer Valid -- Talker lets DAV go high when it sees that all listeners have accepted data (3). Talker may now change the data.
2. Data Valid -- Talker sees all listeners are ready for data (5). It pulls DAV low when the new data is ready.
3. Data Accepted -- The last listener has accepted the data and then released NDAC.
4. Acknowledge Data Not Valid -- The first listener that sees that data is valid (2) pulls NDAC low.
5. Ready For Data -- The last listener has seen that data is no longer valid (1) and has released NRFD to show that it is ready to accept data.
6. Not Ready For Data -- The first listener to accept data after data becomes valid (2) pulls NRFD low.
7. When it sees that data is not valid (1), each listener pulls NDAC low (4) and simultaneously releases NRFD (5).
8. When it sees that data is valid (2), each listener pulls NRFD low (6) and simultaneously releases NDAC.

Figure 2-4. Three-wire Handshaking Protocol.

## **GPIB FUNCTIONS**

The Apple II GPIB has eleven general functions which control devices on the bus and transmit data. Every IEEE-488 compatible device is able to perform at least one of these functions. Please note that not all of the devices you may want to use on the bus will be able to use all of these functions. The general functions are briefly described in the following paragraphs.

### SH1      Source Handshake

The source handshake function uses the three transfer lines to allow a talker (other than the controller) to transfer data to one or more listeners or the Apple on the bus.



#### AH1      Acceptor Handshake

The acceptor handshake function uses the three transfer lines to allow a listener to accept data from a talker which is usually the Apple.

#### T3      BASIC Talker

The controller uses the control and data lines to address a device on the bus and enable the device to be a talker. When a device is enabled as a talker, the device can send status and data bytes on the bus. This function name has no relationship to the BASIC language.

#### L1      BASIC Listener

The controller uses the control and data lines to address a device on the bus and enable the device to be a listener. When a device is enabled as a listener, the device can receive commands and data from the bus. This function name has no relationship to the BASIC language.

#### C4      Respond to Service Request

The service request function uses the SRQ control line to notify the controller (the Apple) of action required.

#### C3      Remote Enable

The remote enable function will enable a device to be controlled by the Apple rather than by the device front panel controls.

#### C25      Parallel Poll

The parallel poll function uses the eight data bus lines to poll up to eight devices on the bus simultaneously. A parallel poll is used to determine whether a device needs to be serviced by the Apple, and if so, which device. The parallel poll is much faster than the service request function, but requires that the Apple conduct frequent parallel polling operations. Many currently available devices do not have parallel poll capabilities.

#### DC      Device Clear

The device clear function permits the controller to clear or initialize (set to a known state) a device on the bus. The device clear function is separate from the interface clear function, which is controlled by the IFC line.

## C2      Interface Clear

The interface clear function is used to set the GPIB interface card controller circuit, and all devices, to a known state prior to use.

## DT      Device Trigger

The device trigger function is used by the controller to synchronize the operation of one or more devices on the bus. The Apple can initiate a group execute trigger (GET) to synchronize a number of devices on the bus. The group execute trigger is not to be confused with the GET instruction in BASIC.

## C1      Controller

The controller function allows the controlling device (the Apple) to send addresses and universal and addressed commands to the devices on the bus. Only one system controller is allowed on the bus. The Apple is the system controller.

## CHAPTER 3

# PROGRAMMING THE GPIB

This chapter contains the programming concepts necessary to write a BASIC program for the GPIB. It begins with a section which explains a little about how the bus works, and how devices generally expect the bus to operate. Next is a summary of the addressing scheme used by the Apple II GPIB, followed by an in-depth description of each command that is available on the GPIB. The final section of the chapter discusses error messages and their causes.

## GETTING STARTED

---

This section briefly explains how to get started programming the IEEE-488 card, some common features of different instruments, and what you should look for in the documentation of the instrument you are interfacing to the bus. It is a general introduction to the bus commands that are described later. All commands referred to in this section will be explained later.

Before you can use the bus, you must issue a PR#n or IN#n command (for the IEEE-488 card in slot n). Every time a PR#n or IN#n is used, the default settings of the card are set to the following values: screen off (SC0), linefeed off (LF0), device number 0 (DV0), and end of string character equal to carriage return (<CR>). If, for example, you do a PR#0 and IN#0 to ask a question of the operator, and then wish to direct I/O back to the card without first resetting the default values, you may use the subroutine in Program 3 of Appendix D.

The format of the output data generally differs from one device to another. Some devices send out a <CR><LF> at the end of a string, while others send out only a <CR>. The appropriate linefeed command, LF1 or LF0, must be used to ensure proper reception of the data. Some devices send out commas or colons as delimiters between strings of data, or between a function code and a data string. Applesoft terminates input of the current string when one of these characters is detected; therefore, for each comma or colon in the input string, the INPUT statement must have an additional variable. If the number of commas or colons sent by the device is variable, you may want to use the subroutine, Input With Commas, in Program 4 of Appendix D.

Some devices must be initialized with certain commands before they can be used for input. Some instruments require a <CR><LF> for setup, and others just need a <CR>. Some devices must be set into "remote" mode before they will respond to commands sent on the IEEE-488 bus, others require "local lockout" mode as well. The commands RA, RM, IA, LL and LO are used to set devices to these different modes.

Certain instruments must receive a trigger command (TG) before they will send a reading to the bus. The data is usually latched into the instrument at the time of the trigger, and held until it is read by some device on the bus. The read operation clears the latch until another trigger command is issued to that device. Some instruments might give false readings if they are not triggered before they are read.

Other commands are used to put the bus and the instruments on it into a known state. These commands are CA, CL, UT and AB. They will reset certain of the instrument's modes, usually those which determine whether the device is set up to receive or to send data.

These are all typical conditions which need to be attended to when a device is being used on the bus. In all cases the instrument's instruction manual will explain exactly which commands the device must receive in order to operate properly.



Because this section deals with commands that you know very little about, we suggest that you re-read it after you have studied the command descriptions later in this chapter.

## COMMAND NOTATION

---

The standard form of an GPIB command is a string of characters; each character is a 7 bit ASCII code as sent by the Apple keyboard. Each character is sent or received in the low seven bits of one byte. The general format of a command string is

`<command> <control characters> <data>`

The `<command>` is a two letter mnemonic device used to execute one of the GPIB commands. The `<control characters>` and `<data>` parameters are required by some of the commands, and not required by others.

The commands used by the GPIB are summarized in Table 3-1. The left column lists the command names; the column labelled FORMAT lists the form of each command string in Backus-Naur Form (BNF); and the column labelled BASIC lists the BASIC command (PRINT or INPUT) that must be used to send the command.

All GPIB commands are sent by BASIC to the IEEE-488 card. Depending on the command sent, the card may or may not transfer data on the bus. The rightmost column of Table 3-1, labelled BUS?, says YES if the command in that row of the table causes data to be placed on or read from the bus.

BNF descriptions of the `<control characters>` and `<data>` portions of the command strings are given in Table 3-2. Every bracketed component, called a descriptor, is defined in Table 3-2 in terms of ASCII characters and other descriptors.

In BNF notation, a symbol that is not enclosed in brackets should be included in the command string literally. A descriptor enclosed in brackets is used to represent an element, or list of elements, that must occupy that position in the command string. Descriptors that are separated by a vertical bar (|) are interchangeable. A vertical bar may be read as the English word "or".



For clarity, elements of the command string are separated by spaces, both in the tables and in the individual command descriptions later in the chapter. Actual command strings must not contain these spaces since a space can be interpreted as the address of a listener.

In Table 3-2, note especially that the <listen list> and <count> parameters must be terminated by the character <CTRL-Z>. This may not be obvious from Table 3-1. Note also the use of the symbols <CTRL-Z>, <CR>, and <SP>. These symbols refer to pressing Z while holding down the key labelled CTRL, pressing the key labelled RETURN, and pressing the spacebar, respectively.

Tables 3-1 and 3-2 are also included in the reference card in the back of the manual.

Table 3-1. GPIB Command Summary.

<u>COMMAND</u>	<u>FORMAT</u>	<u>BASIC</u>	<u>BUS?</u>
WRITE	WT <listen list> <data> <EOS>	PRINT	YES
WRITECNT	WC <listen list> <count> <data>	PRINT	YES
READ	RD <talker> <CTRL-Z> <data> <term>	INPUT	YES
READCNT	RC <talker> <CTRL-Z> <count> <data> <term>	INPUT	YES
XFER	XF <talker> <CTRL-Z> <listen list> <CR>	PRINT	YES
TRIGR	TG <listen list> <CR>	PRINT	YES
CLRAL	CA <CR>	PRINT	YES
CLEAR	CL <listen list> <CR>	PRINT	YES
REMAI	RA <CR>	PRINT	YES
REMDV	RM <listen list> <CR>	PRINT	YES
LLKAL	LL <CR>	PRINT	YES
LOCAL	LA <CR>	PRINT	YES
LOCDV	LO <listen list> <CR>	PRINT	YES
SRQD	SR <srq> <CR>	INPUT	YES
SPOLL	SP <talker> <CTRL-Z> <status word> <CR>	INPUT	YES
PPOLL	PP <status word> <CR>	INPUT	YES
PPENB	PE <enable list> <CR>	PRINT	YES
PPDIS	PD <listen list> <CR>	PRINT	YES
PPUAL	PU <CR>	PRINT	YES
ABORT	AB <CR>	PRINT	YES
LINEFEED	LF <off/on> <CR>	PRINT	NO
EOS	ES <EOS> <CR>	PRINT	NO
SCREEN	SC <off/on> <CR>	PRINT	NO
DEVICE	DV <device number>	PRINT	NO
UNTALK	UT <CR>	PRINT	YES

Table 3-2. GPIB Command String Definitions.

```

<listen list> ::= <CTRL-Z> | <listener> <CTRL-Z> | <listener> <listen list>
<listener> ::= <primary listener> | <primary listener> <secondary address>
* <primary listener> ::= <SP> | ! | " | # ... > | ?
* <secondary address> ::= ' | a | b | c ... } | ~
<count> ::= <decimal digit> <CTRL-Z> | <decimal digit> <count>
      NOTE: The maximum total value for count is 255.
<EOS> ::= any ASCII character
<data> ::= empty | any ASCII character | any ASCII character <data>
<talker> ::= empty | <primary talker> | <primary talker> <secondary address>
* <primary talker> ::= @ | A | B | C ... ^ | _
* <hex digit> ::= 0 | 1 | 2 ... 9 | A | B ... F
* <decimal digit> ::= 0 | 1 | 2 ... 9
<status word> ::= <hex digit> <hex digit>
<srq> ::= ASCII "T" | ASCII "F"
<term> ::= empty | <CR> | <EOS> | EOF
      NOTE: empty if last char = <CR> or <CR><LF>
<device number> ::= <decimal digit> <decimal digit>
      NOTE: Maximum device number is 30
<off/on> ::= ASCII "0" | ASCII "1"
<enable list> ::= <listener> <enable> | <listener> <enable> <enable list>
* <enable> ::= @ | A | B ... N | O
<CR> ::= ASCII carriage return (13 decimal)
<LF> ::= ASCII linefeed (10 decimal)
<SP> ::= ASCII space (32 decimal)
<CTRL-Z> ::= ASCII SUB (26 decimal)
<CTRL-D> ::= ASCII ET (4 decimal)

```

\* NOTE: Marked elements are ASCII characters.



# ADDRESSES

---

There are three types of addresses which may be included as part of the <control characters> parameter. These address types are: listener address, talker address and secondary address. Each address is specified by a distinct character corresponding to one of the ASCII codes. Address characters referenced in the following text are enclosed in quotation marks.

## LISTENER ADDRESSES

The addresses of all listeners on the GPIB are restricted to the hex values 20 through 3E. The listener address range corresponds to ASCII characters (as sent from the keyboard) from <SP> through ">". The decimal, octal, and hexadecimal values of each possible listener address are shown in Table A-1 of Appendix A. Hex value 3F (ASCII "?") is reserved as a special universal UNListen (UNL) command which is automatically sent by the Apple II GPIB when needed.

## TALKER ADDRESSES

There may only be a single talker at a time. The address of the talker on the GPIB is restricted to the hex values 40 to 5E. The talker address range corresponds to ASCII characters (as sent from the keyboard) from "@" through "~". The decimal, octal, and hexadecimal values of each possible talker address are given in Table A-2 of Appendix A. Hex value 5F (ASCII "\_") is reserved as a special universal UnTalk command which is sent with the UT command.

## SECONDARY ADDRESSES

Secondary addresses may be used to set operating features of some devices. They may also be used to address subunits of a complex device. For instance, the operating range and measurement type for some remote digital voltmeters may be set from the Apple by means of secondary addresses.

The GPIB provides the capability of sending a secondary address to either talkers or listeners. Secondary addresses are included as elements of the <talker> or <listen list> parameters if they are needed. A secondary address must immediately follow the primary address with which it is associated.

Secondary addresses on the GPIB are restricted to the hex values 60 through 7E. The secondary address range corresponds to ASCII characters (as sent from the keyboard) from "'" through "~". The decimal, octal, and hexadecimal values of each possible secondary address are shown in Table A-3 of Appendix A.

From APPLESOFT, the characters needed for secondary addresses are not directly available from the keyboard. They can, however, be generated using the CHR\$ function. The proper CHR\$ calls are shown in Table A-3 of Appendix A. Examples of the use of this function are given in the examples of the commands.

From Integer BASIC, a special subroutine must be used to generate secondary addresses. This subroutine is listed in Program 1 of Appendix D.

## ON USING ADDRESSES

You must carefully read the operating manual of the device you are planning to use with the GPIB, to ensure that you understand the addressing method used by that device. The device's addresses might be preset by factory-installed jumpers, or they might be adjustable using addressing switches. It is important that you determine what address values the manufacturer intended the device to have.

The address for the device may be given in binary, hex, octal, or ASCII. You must determine what character, when sent from the Apple keyboard, will represent that address. If the address is not hard-wired, you should set the address switches or jumpers to a convenient address. Tables of addresses are given in Appendix A.

Each device has associated with it a device number assigned by the manufacturer or set as explained above. The device number is only used internally to the IEEE-488 card.



Be sure that no instrument has the same device number as the Apple. The default device number for the Apple is 0. Device number 0 (not ASCII 0) corresponds to a listener address <SP> and a talker address "@".

## GPIB COMMAND DESCRIPTIONS

---

Table 3-1 lists all the commands that you can send to the Apple II GPIB. There are three types of commands: output commands, input commands, and commands used to set operating characteristics of the IEEE-488 card. The last type does not cause any data to be placed on the bus; the others do. The two rightmost columns of Table 3-1 indicate which type each command is. The column labelled BASIC tells which BASIC command, PRINT or INPUT, should be used to send the command in each row.

### SENDING A COMMAND

Commands which send data to a device take the form

```
PRINT "<command><control chars><data>"
```

and commands which read data from a device take the form

```
INPUT "<command><control chars>"; S$
```

or

```
PRINT "<command><control chars>";  
INPUT S$
```

in which S\$ is a string appropriate for that particular command.

A program that sends commands to the GPIB must first indicate the number of the slot into which the interface card is inserted. In the examples which follow, the slot number is assumed to be 3. The statement

PR#3

must precede the first statement in which you PRINT a command to the card. Likewise the statement

IN#3

must precede the first statement in which you INPUT data from the card. Quite simply, you are telling the Apple to talk to or listen to the Apple II GPIB, which is in slot 3 (in this case).



A DOS command from a program must appear in a PRINT statement whose first output character is <CTRL-D> and whose output is separated from preceding and from succeeding printed output by <CR>s.

Thus, if you are using DOS, the statements

```
10 D$ = "": REM A <CTRL-D> LIES BETWEEN THE QUOTES
20 PRINT D$;"PR#3"
30 PRINT D$;"IN#3"
```

must precede the first PRINT and INPUT statements to the card. The DOS form of the commands is the one that we will use in the following examples.



The commands PR#n and IN#n have the effect of resetting all the card's default values. The resetting of default values can be avoided with the subroutine in Program 3 of Appendix D.

## COMMAND PROTOCOL

As previously noted, <listen list> and <count> must be terminated by the character <CTRL-Z>. For the following examples, assume that the variable Z\$ has been set to the value <CTRL-Z> using one of the BASIC statements

```
5 Z$ = "": REM <CTRL-Z> IS BETWEEN THE QUOTES
5 Z$ = CHR$(26) : REM CHR$(26) IS <CTRL-Z> IN APPLESOFT
```

and that D\$ has been set to the value <CTRL-D> as shown above.

When using the GPIB with the SCREEN option set to ON, a <CTRL-Z> prints out as an inverse-video Z. See the SCREEN command later in this chapter for more details.

A command string is terminated by a <CR> or an end-of-string character, <EOS>. The PRINT command normally sends a <CR> as its last character. If <EOS> is not <CR>, or if you want to PRINT a string containing characters as well as variables, use the special symbol ";" to continue the PRINT statement without sending a <CR>. For example, to print the character sequence "WT1" followed by <CTRL-Z> and then A\$, without <CR>s between them, use

```
10 PRINT "WT1" ; Z$ ; A$ : REM THE VALUE OF Z$ IS <CTRL-Z>
```

This is equivalent to the statements

```
10 PRINT "WT1";  
11 PRINT Z$;  
12 PRINT A$
```

It is, of course, possible to embed control characters into any string, but since embedded control characters are not visible, the above convention was adopted.

## BUS TRANSFERS

When a command is executed by the GPIB, a sequence of data and control signals is transferred on the 16 data and control lines. Examples of the input/output sequences for all the GPIB commands are given in Appendix B. The bytes sent on the DIO lines are data and IEEE-488 commands. Refer to the IEEE standard for more details.

## GPIB COMMANDS

---

The first four commands given below -- LINEFEED, EOS, SCREEN and DEVICE -- are commands that set operating characteristics of the card. Although these commands do not cause any data to be sent on the bus, some of them do have an effect on data that is sent over the bus by other commands.

The rest of the commands cause some sort of data to be transferred over the bus: this data can take the form of information passed from one device to another, or it can take the form of commands that set operating characteristics of the devices on the bus.

### LINEFEED: LF

Linefeed Control. The LINEFEED command is used to set the board to a mode that will generate a line feed character (<LF>) after each carriage return (<CR>). The LINEFEED command takes the form

LF <off = 0/on = 1> <CR>

During WRITE commands issued while LINEFEED is set to on, the IEEE-488 card will send a <LF> following every instance of <CR> in

<data>, even if <data> already has a <LF> after each <CR>. The LINEFEED option does not affect the data sent by WRITECNT commands. During READ commands issued while LINEFEED is set to on, the IEEE-488 card expects a <LF> to follow the terminating <CR>.

The LINEFEED commands are as follows:

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "LF1" (line feed character is provided after each <CR>)

20 PRINT "LF0" (line feed character is not provided after each <CR>)
```

LINEFEED is set to off when <EOS> is changed from <CR>.

## EOS: ES

End Of String Character. The EOS command is used to change the character used to terminate a string. The EOS character is a <CR> under normal or default conditions. A reset or PR#3 instruction resets the EOS character to <CR>. LINEFEED is set to off when <EOS> is changed from <CR>. The EOS command is

```
ES <EOS> <CR>
```

The EOS command format is as follows:

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "ES!"
```

This command changes the EOS character from <CR> to !.

## SCREEN: SC

Screen Control. The SCREEN command is used to control the CRT display during operation of the GPIB. The CRT display shows all input and output characters when SCREEN control is on. The display is very useful during program debugging. The normal (default) condition of SCREEN is off. The SCREEN command takes the form

```
SC <off = 0/on = 1> <CR>
```

The SCREEN command is as follows:

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "SC1" (data is displayed on the CRT display)

20 PRINT "SC0" (data is not displayed on the CRT)
```

## DEVICE: DV

Controller Device Number. The DEVICE command is used to set the device number of the Apple controller. The normal or default device number of the Apple controller is 0, and it will be reset to 0 by a PR#n instruction, or by pressing RESET. The device number, used modulo 32, may be one or two ASCII decimal digits; if the number is 09 or less, the leading 0 can be omitted. The device number 31 (corresponding to 31,63,95...) may not be used because it is reserved for the UNLISTEN and UNTALK commands. The DEVICE command takes the form

DV <device number> <CR>

The DEVICE command follows:

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "DV01"
```

## WRITE: WT

Write Data Out. The WRITE command sends data to all devices named in <listen list>. The WRITE command is the normal way of sending data from the Apple to one or more remote devices. The command takes the form

WT <listen list> <data> <EOS>

An EOI (End-Or-Identify) bus signal is automatically sent with the last <EOS> character. A <CR> is the normal <EOS>. If the LF flag is set (see the LF command), and the <EOS> character is <CR>, then EOI is sent with the <LF>. Sample WRITE commands follow:

```
10 D$;"PR#3": REM THE VALUE OF D$ IS <CTRL-D>
30 PRINT "WT1"; Z$; A$ : REM THE VALUE OF Z$ IS <CTRL-Z>
```

This command sends the string of data, A\$, to the listener with address equal to the value of keyboard character 1 (49 decimal, 61 octal or 31 hex).

```
40 PRINT "WT8"; CHR$(97); Z$; A$ : REM THE VALUE OF Z$ IS <CTRL-Z>
```

This command sends the string of data, A\$, to the listener with primary address "8" and secondary address "a". The CHR\$ function must be used to send lower-case characters such as "a" from the Apple keyboard. The CHR\$ functions are listed in table A-3 of Appendix A. The ASCII code for "a" is 97, so CHR\$(97) generates an "a".

Program 1 in Appendix D shows how lower case characters can be sent from Integer BASIC.

## WRITECNT: WC

Write Data Out With Count. The WRITECNT command is like a WRITE command that includes a count of the number of bytes in the data string. The command takes the form

```
WC <listen list> <count> <data>
```

The <EOS> character (nominally <CR>) is not normally sent at the end of the data string, but the bus signal EOI is sent with the last character. The count must be greater than 0 and less than 256. A sample WRITECNT statement follows:

```
10 D$;"PR#3":REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "WC1"; Z$; LEN(B$); Z$; B$; :REM THE VALUE OF Z$ IS <CTRL-Z>
```

This command sends the string B\$ of length LEN(B\$) to the listener with address "1".



One important use of the WRITECOUNT command is to send data that is not terminated by the <EOS> character. This cannot be done with the WRITE command.



Any extra characters beyond the count will be interpreted as a new command and will cause an error. Too few characters will cause the next command to be sent as part of the data string and will cause an error. If the PRINT statement is not terminated by a ";", a <CR> will be automatically sent and must be included in the character count. Likewise, if LINEFEED is set, <CR><LF> must be included in the count.

## READ: RD

Read Data In. The READ command receives data from a talker specified by the command. The READ command is the normal method of receiving data from a remote device. The command takes the form

```
RD <talker> Z$ <data> <term>
```

The input data string terminates when a <CR> or <EOS> character (nominally a <CR>) is received, or when the EOI signal goes true. The IEEE-488 card will always return a <CR> to the program to terminate the input data. If the last character from the card is a <CR>, an additional <CR> will not be sent. A line feed character from the GPIB will be suppressed and not passed to the program. A sample READ command follows:

```
5 D$;"PR#3":REM THE VALUE OF D$ IS <CTRL-D>
10 D$;"IN#3":REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "RDA"; Z$; :REM THE VALUE OF Z$ IS <CTRL-Z>
25 INPUT C$
```



This command receives the data string C\$ from the talker with address "A".

```
5   D$;"PR#3" :REM  THE VALUE OF D$ IS <CTRL-D>
10  D$;"IN#3" :REM  THE VALUE OF D$ IS <CTRL-D>
30  PRINT "RDA"; CHR$(98); Z$; : REM  THE VALUE OF Z$ IS <CTRL-Z>
35  INPUT C$
```

This command receives a data string C\$ from the talker with address "A" and secondary address "b". The lower case "b" is generated by CHR\$(98).



Note the use of the ";" following the last two PRINT statements. If a PRINT statement sends a command and an INPUT statement reads the data placed on the bus because of that command, it is necessary to suppress the <CR> normally sent after the PRINT.

## READCNT: RC

Read Data In With Count. The READCNT command is like the READ command with a field, count, that specifies the length of the input string in bytes. The command takes the form

RC <talker> Z\$ <count> <data> <term>

EOI may be sent with the last character in the string by the talker. The data string may contain less than count characters if the talker uses EOI or <EOS>.

The IEEE-488 card will always return a <CR> to the program to terminate the input data. If the last character from the card is a <CR>, an additional <CR> will not be sent. An example of a READCNT command follows:

```
9   D$;"PR#3" :REM  THE VALUE OF D$ IS <CTRL-D>
10  D$;"IN#3" :REM  THE VALUE OF D$ IS <CTRL-D>
20  PRINT "RCB"; Z$; "26"; Z$; : REM  THE VALUE OF Z$ IS <CTRL-Z>
25  INPUT C$
```

This command receives the data string C\$, containing 26 characters, from the talker with address "B".



If you are using the RC or RD command with the LINEFEED option set (see the LF command), be sure that the device does not send a NULL (ASCII 00) when the GPIB is expecting a <LF>.

## XFER: XF

Transfer Data. The XFER command is used to transfer data between remote devices other than the Apple. The talker is addressed first, followed by the listeners. The device address of the Apple (nominally 0) must NOT appear as the talker or in the listen list. The XFER command takes the form

XF <talker> Z\$ <listen list> <CR>

The talker MUST be programmed to send EOI with the last character in the transfer. This tells the listener(s) that the transfer is complete, and allows the Apple to regain control of the GPIB. A sample XFER command follows:

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "XFA"; Z$; "1"; Z$
```

This command allows the talker with address "A" to transfer data directly to the listener with address "1".

## TRIGR: TG

Group Execute Trigger. The TRIGR command is used to configure or set up a group of remote devices on the GPIB and to cause all of the devices to commence execution at the same time. The command takes the form

TG <listen list> <CR>

A sample TRIGR command follows:

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "TG"; "19"; Z$
```

This command causes listeners with addresses "1" and "9" to be triggered simultaneously.

## CLRAL: CA

Clear All Devices. The CLRAL command issues a universal clear message to all remote devices on the GPIB. The command format is

CA <CR>

An example of the CLRAL command is

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "CA"
```

This command clears all devices on the GPIB.

## CLEAR: CL

Clear Selected Devices. The CLEAR command clears the remote devices specified by the address list. The command takes the form

```
CL <listen list> <CR>
```

A sample CLEAR command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "CL"; "19"; Z$
```

This command issues a clear message to the remote devices with addresses "1" and "9".

## REMAI: RA

Remote Enable All. The REMAI command is used to set the REN (remote enable) control line true. The remote devices will not be in remote operating mode until addressed to listen. The REMAI command is

```
RA <CR>
```

An example of a REMAI command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "RA"
```



All subsequent listen addresses will be sent with the REN signal true. The REN signal is not set to false until the LA command is sent.

## REMDV: RM

Remote Enable Selected Devices. The REMDV command is used to set the REN (remote enable) line true and to send the address of the specified listeners on the GPIB. The listeners are placed in remote mode when addressed. The command takes the form

```
RM <listen list> <CR>
```

A sample REMDV command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
15 E$ = "12349" + Z$
2Ø PRINT "RM"; E$
```

This command turns on remote operation of the devices with addresses "1", "2", "3", "4" and "9".



All subsequent listen addresses will be sent with the REN signal true. The REN signal is not set to false until the LA command is sent.

## LLKAL: LL

Local Lockout All Devices. The LLKAL command is used to set all remotod devices on the GPIB to local lockout mode. The front panel controls of a remote device have no effect (e.g. if someone presses the instrument's RESET button) when the device is in local lockout mode. The LLKAL command follows:

LL <CR>

Here is an example of the LLKAL command

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "LL"
```

## LOCAL: LA

Local Mode All Devices. The LOCAL command is used to set all remote devices on the GPIB to local operating mode. The command byte takes the form

LA <CR>

The LOCAL command will set the REN control line false. A LOCAL command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "LA"
```

## LOCDV: LO

Local Mode Selected Devices. The LOCDV command is used to set selected remote devices to local operating mode. The LOCDV command takes the form

LO <listen list> <CR>

A sample LOCDV command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "LO"; "12349"; Z$
```

This command sets local operating mode for the remote devices with addresses "1", "2", "3", "4" and "9".

## SRQD: SR

Service Requested. The SRQD command is used to determine whether any remote device on the GPIB is requesting service. The command will return the value of ASCII T (84 decimal, 124 octal, 54 hex) if the SRQ control line is true. The command will return the value of ASCII F (70 decimal, 106 octal, 46 hex) if the SRQ control line is false. The SRQD command takes the form

```
SR <srq> <CR>
```

The SRQD command follows:

```
5   PRINT D$;"PR#3"   :REM THE VALUE OF D$ IS <CTRL-D>
10  PRINT D$;"IN#3"   :REM THE VALUE OF D$ IS <CTRL-D>
20  INPUT "SR"; G$
```

The value returned in G\$ is either a "T" or a "F".

## SPOLL: SP

Serial Poll. The SPOLL command is used to send a serial poll to an addressed remote device on the GPIB. The SPOLL is usually used when a "T" has been returned as the result of an SRQD command. The addressed remote device will return a status byte which will determine if that device has requested service. The IEEE-488 card translates the status byte into a 2 byte status word. The form of the SPOLL command is

```
SP <talker> Z$ <status word> <CR>
```

A sample SPOLL command follows:

```
5   PRINT D$;"PR#3"   :REM THE VALUE OF D$ IS <CTRL-D>
10  PRINT D$;"IN#3"   :REM THE VALUE OF D$ IS <CTRL-D>
20  PRINT "SPA"; Z$;
30  INPUT H$
```

The value returned in H\$ is a status word from the remote device with address "A". H\$ is two bytes long. The first byte is the ASCII form of the most significant four bits of the binary status byte, representing a hexadecimal number between 0 and F. The second byte is the ASCII form of the least significant four bits of the binary status byte. <CTRL-Z> will not be displayed on the screen during SPOLL command. The value of the status byte returned by a device is determined by the manufacturer of that device.

If the addressed device was asserting SRQ, it will stop doing so and it will return the status byte with DIO7 asserted. This implies that the first character of H\$ will be a 4, 5, 6, 7, C, D, E, or F. The meaning of other DIO bits is device dependent.

## PPOLL: PP

Parallel Poll. The PPOLL command is used to conduct a parallel poll of remote devices. The result of the parallel poll is to return an eight bit status byte to the IEEE card. Each of the eight bits may be assigned to a device using the PPENB command for some devices, or by a DIP switch on devices that implement the PP2 subset of the IEEE-488 standard. The IEEE-488 card translates the status byte into a 2 byte status word. The form of the PPOLL command is

PP <status word> <CR>

A sample PPOLL command follows:

```
5 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
10 PRINT D$;"IN#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 INPUT "PP"; I$
```

The value returned in I\$ is a two byte status word. The first byte is the ASCII form of the most significant four bits of the binary status byte, representing a hexadecimal number between 0 and F. The second byte is the ASCII form of the least significant four bits of the binary status byte.

## PPENB: PE

Parallel Poll Enable. The PPENB command is used to set the configuration of remote devices which can implement the PP1 subset of the IEEE-488 standard. The command is used to assign a bit number and bit sense for each remote device to be enabled.

PE <enable list> <CR>

An element of the enable list consists of a listener address followed by an enable byte. The enable byte takes the form 0 1 0 0 S P<sub>3</sub>P<sub>2</sub>P<sub>1</sub>. Bit "S" of the enable byte assigns the sense of that listener's poll bit in the status byte (see above); that is, whether the device will signal a request for service with a 1 or a 0. Bits P<sub>3</sub>P<sub>2</sub>P<sub>1</sub>, treated as a number from 0 through 7, determine which bit of the status byte is assigned to that device. The enable byte corresponds to an ASCII character in the range "Q" to "O". To conform with the IEEE standard, the IEEE-488 card ORs the status byte with a hexadecimal 20 before sending the byte on the bus. A sample PPENB command follows:

```
10 PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
20 PRINT "PE"; "1A2B3K4L"
```

The sample command results in the following configuration

```
Listener "1" signals requests with status bit 1, sense = 0
Listener "2" signals requests with status bit 2, sense = 0
Listener "3" signals requests with status bit 3, sense = 1
Listener "4" signals requests with status bit 4, sense = 1
```

## PPDIS: PD

Parallel Poll Disable. The parallel poll disable is used to prevent the listed remote devices from responding to parallel poll commands. The PPDIS command takes the form

```
PD <listen list> <CR>
```

A sample PPDIS command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "PD"; "1234"; Z$
```

This command prevents the listeners with addresses "1", "2", "3" and "4" from responding to parallel poll commands.

## PPUAL: PU

Parallel Poll Unconfigure All Devices. The PPUAL command is used to change the configuration of remote devices in order to prevent future responses to parallel polls. The PPUAL command is

```
PU <CR>
```

The PPUAL command is shown:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "PU"
```

## ABORT: AB

Clear All Interfaces. The ABORT command is used only in cases of emergency to clear all device interfaces. It resets all devices to a known state by setting the IFC control line to true. The IFC pulse lasts for about 5 milliseconds. The ABORT command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "AB"
```

## UNTALK: UT

Untalk. The UNTALK is used to send the Universal Untalk (UNT) command. This command is required for some remote devices. Carefully read the device's operating manual for actual requirements.

The UNTALK command follows:

```
1Ø PRINT D$;"PR#3" :REM THE VALUE OF D$ IS <CTRL-D>
2Ø PRINT "UT"
```



# GPIB ERROR MESSAGES

---

When the GPIB encounters a stream of characters which it cannot understand, it issues an error message to the console. Execution of the program is terminated unless the BUS error occurs when the ONERR flag is set. There are four different error messages which are listed below along with typical causes of that particular error.

When displayed on your screen, the three letter error type is followed by the letters "ERR".

CMD -- The first two characters of the command were not recognized as one of the GPIB commands. This could happen accidentally if, for example, <EOS> is no longer <CR> and a previous WRITE command didn't use a semicolon to suppress the final <CR>. It could also be caused by an unexpected comma, causing termination of input, in an input string. Another cause of the CMD error is the occurrence of a system error during an input command. The system error message starts with a "?" which is interpreted as an invalid command character.

LST -- The control characters and/or data do not follow the protocol required by the command being used. This error is typically caused by an omitted <CTRL-Z> or consecutive secondary addresses.

CNT -- The count is out of range (usually  $\geq 256$ ). This error can only occur with the WC, RC and DV commands.

BUS -- Incomplete source handshake error. It occurs if a byte is placed on the GPIB when NDAC and NRFD are both high. This situation will occur when the bus is improperly connected or if the addressed device is off. A BUS error can be recovered using the ONERR...GOTO statement.

## THE ONERR . . . GOTO STATEMENT

Of the four error conditions that the GPIB can detect, only the BUS error is recoverable. If you are using APPLESOFT and your program includes an

ONERR GOTO linenum

statement, the GPIB firmware will detect the ONERR flag, and then transfer control to the proper place in the APPLESOFT firmware. APPLESOFT will then transfer control to linenum, as if a normal APPLESOFT error had occurred. You may insert any type of error recovery routine at the line indicated by linenum. A sample error routine is listed in Program 2 of Appendix D.

See the APPLESOFT II BASIC Programming Reference Manual for more details on the ONERR...GOTO statement.

## APPENDIX A

# ASCII CONVERSION TABLES

DEC = ASCII decimal code

OCT = ASCII octal code

HEX = ASCII hexadecimal code

CHR\$ = Applesoft function

### TABLE A-1 GPIB LISTENER ADDRESSES

DEC	OCT	HEX	CHAR	WHAT TO TYPE
32	040	20	SPACE	space
33	041	21	!	!
34	042	22	"	"
35	043	23	#	#
36	044	24	\$	\$
37	045	25	%	%
38	046	26	&	&
39	047	27	'	'
40	050	28	(	(
41	051	29	)	)
42	052	2A	*	*
43	053	2B	+	+
44	054	2C	,	,
45	055	2D	-	-
46	056	2E	.	.
47	057	2F	/	/
48	060	30	0	0
49	061	31	1	1
50	062	32	2	2
51	063	33	3	3
52	064	34	4	4
53	065	35	5	5
54	066	36	6	6
55	067	37	7	7
56	070	38	8	8
57	071	39	9	9
58	072	3A	:	:
59	073	3B	;	;
60	074	3C	<	<
61	075	3D	=	=
62	076	3E	>	>
63	077	3F	?	Reserved for UNLISTEN

# TABLE A-2 GPIB TALKER ADDRESSES

<u>DEC</u>	<u>OCT</u>	<u>HEX</u>	<u>CHAR</u>	<u>WHAT TO TYPE</u>
64	100	40	@	@
65	101	41	A	A
66	102	42	B	B
67	103	43	C	C
68	104	44	D	D
69	105	45	E	E
70	106	46	F	F
71	107	47	G	G
72	110	48	H	H
73	111	49	I	I
74	112	4A	J	J
75	113	4B	K	K
76	114	4C	L	L
77	115	4D	M	M
78	116	4E	N	N
79	117	4F	O	O
80	120	50	P	P
81	121	51	Q	Q
82	122	52	R	R
83	123	53	S	S
84	124	54	T	T
85	125	55	U	U
86	126	56	V	V
87	127	57	W	W
88	130	58	X	X
89	131	59	Y	Y
90	132	5A	Z	Z
91	133	5B	[	CHR\$(91)
92	134	5C	\	CHR\$(92)
93	135	5D	]	] (shift-M)
94	136	5E	^	^
95	137	5F	—	Reserved for UNTALK

# TABLE A-3 GPIB SECONDARY ADDRESSES

<u>DEC</u>	<u>OCT</u>	<u>HEX</u>	<u>CHAR</u>	<u>WHAT TO TYPE</u>
96	140	60	'	CHR\$(96)
97	141	61	a	CHR\$(97)
98	142	62	b	CHR\$(98)
99	143	63	c	CHR\$(99)
100	144	64	d	CHR\$(100)
101	145	65	e	CHR\$(101)
102	146	66	f	CHR\$(102)
103	147	67	g	CHR\$(103)
104	150	68	h	CHR\$(104)
105	151	69	i	CHR\$(105)
106	152	6A	j	CHR\$(106)
107	153	6B	k	CHR\$(107)
108	154	6C	l	CHR\$(108)
109	155	6D	m	CHR\$(109)
110	156	6E	n	CHR\$(110)
111	157	6F	o	CHR\$(111)
112	160	70	p	CHR\$(112)
113	161	71	q	CHR\$(113)
114	162	72	r	CHR\$(114)
115	163	73	s	CHR\$(115)
116	164	74	t	CHR\$(116)
117	165	75	u	CHR\$(117)
118	166	76	v	CHR\$(118)
119	167	77	w	CHR\$(119)
120	170	78	x	CHR\$(120)
121	171	79	y	CHR\$(121)
122	172	7A	z	CHR\$(122)
123	173	7B	{	CHR\$(123)
124	174	7C		CHR\$(124)
125	175	7D	}	CHR\$(125)
126	176	7E	~	CHR\$(126)

# TABLE A-4 ASCII CONTROL CODES

<u>DEC</u>	<u>OCT</u>	<u>HEX</u>	<u>CHAR</u>	<u>WHAT TO TYPE</u>
0	000	00	NULL	CTRL-@
1	001	01	SOH	CTRL-A
2	002	02	STX	CTRL-B
3	003	03	ETX	CTRL-C
4	004	04	ET	CTRL-D
5	005	05	ENQ	CTRL-E
6	006	06	ACK	CTRL-F
7	007	07	BEL	CTRL-G
8	010	08	BS	CHR\$(8)
9	011	09	HT	CTRL-I
10	012	0A	LF	CTRL-J
11	013	0B	VT	CTRL-K
12	014	0C	FF	CTRL-L
13	015	0D	CR	CTRL-M or RETURN
14	016	0E	SO	CTRL-N
15	017	0F	SI	CTRL-O
16	020	10	DLE	CTRL-P
17	021	11	DC1	CTRL-Q
18	022	12	DC2	CTRL-R
19	023	13	DC3	CTRL-S
20	024	14	DC4	CTRL-T
21	025	15	NAK	CHR\$(21)
22	026	16	SYN	CTRL-V
23	027	17	ETB	CTRL-W
24	030	18	CAN	CTRL-X
25	031	19	EM	CTRL-Y
26	032	1A	SUB	CTRL-Z
27	033	1B	ESCAPE	ESC
28	034	1C	FS	CHR\$(28)
29	035	1D	GS	CTRL-SHIFT-M
30	036	1E	RS	CTRL-SHIFT-N
31	037	1F	US	CHR\$(31)

## APPENDIX B

# BUS INPUT/OUTPUT SEQUENCES

This appendix illustrates the input/output sequences for the GPIB with an example of each command. The DIO mnemonics are messages defined by the IEEE-488 standard. Here are the meanings of the messages that the Apple uses. For more information see the IEEE-488 standard.

<u>DIO Command</u>	<u>Hex</u>	<u>Binary</u>	<u>Translation</u>
DCL	14	X0010100	Device CLear
GET	08	X0001000	Group Execute Trigger
GTL	01	X0000001	Go To Local
LLO	11	X0010001	Local LockOut
MLA		X01 Dev.Adr.	My (the Apple) Listen Address
MTA		X10 Dev.Adr.	My (the Apple) Talk Address
PPC	05	X0000101	Parallel Poll Configure
PPD	70	X1110000	Parallel Poll Disable
PPE	60	X110SPPP	Parallel Poll Enable
PPU	15	X0010101	Parallel Poll Unconfigure
SDC	04	X0000100	Select Device Clear
SPD	19	X0011001	Serial Poll Disable
SPE	18	X0011000	Serial Poll Enable
UNL	3F	X0111111	UNListen
UNT	5F	X1011111	UNTalk

MLA and TLA are formed using the 5 lowest bits of the Apple's device address. Three mnemonics, LSTN, TALK, and XFER, are used which are not derived from the IEEE-488 standard.

**LSTN** — The controller is waiting to carry out a transaction with a listener. The Apple will not send new data (signalled by pulling the DAV line low) until all listeners are ready to accept data (NRFD = 1) and one listener has acknowledged the fact that there is no data currently valid (by pulling the NDAC line low).

**TALK** — The controller is waiting to carry out a transaction with a talker. The Apple will not accept data (by setting NDAC to 1) until the data is valid (DAV = 0) and one listener has recognized that there is valid data on the bus (by pulling NRFD low).

**XFER** -- The addressed talker and listeners will carry out a handshake sequence to transfer data on the bus. The Apple does not participate.

In the commands on the next page, the character sequence <CR> represents pressing the key labelled RETURN (ASCII 13). It is not to be interpreted character by character. As before, Z\$ is a variable that has been set to contain the value <CTRL-Z> (ASCII 26).

<u>COMMAND</u>	<u>DIO</u> <u>MNEMONIC</u>	<u>DIO</u> <u>(HEX)</u>	<u>MGMNT</u>	<u>HNDSHK</u>
PRINT "WT1";Z\$;"DATA"				
or				
PRINT "WC1";Z\$;"5";Z\$;"DATA<CR>";				
	MTA	40	ATN	LSTN
	UNL	3F	ATN	LSTN
primary listener		31	ATN	LSTN
D		44		LSTN
A		41		LSTN
T		54		LSTN
A		41		LSTN
<CR> [EOS]		0D	EOI	LSTN
idle		XX	ATN	
PRINT "RDAa";Z\$;				
INPUT C\$				
or				
PRINT "RCAa";Z\$;"5";Z\$;				
INPUT C\$				
	primary talker	41	ATN	LSTN
	secondary talker	61	ATN	LSTN
	UNL	3F	ATN	LSTN
	MLA	20	ATN	LSTN
	D	44		TALK
	A	41		TALK
	T	54		TALK
	A	41		TALK
	<CR>	0D	[EOI]	TALK
	idle	XX	ATN	
PRINT "XFA";Z\$;"19";Z\$				
	primary talker	41	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	primary listener	39	ATN	LSTN
	D	44		XFER
	A	41		XFER
	T	54		XFER
	A	41	EOI	XFER
	idle	XX	ATN	
PRINT "TG1";Z\$				
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	GET	08	ATN	LSTN
	idle	XX	ATN	



<u>COMMAND</u>	<u>DIO MNEMONIC</u>	<u>DIO (HEX)</u>	<u>MGMNT</u>	<u>HNDSHK</u>
PRINT "CA"	DCL idle	14 XX	ATN ATN	LSTN
PRINT "CL1";Z\$	UNL primary listener SDC idle	3F 31 04 XX	ATN ATN ATN ATN	LSTN LSTN LSTN LSTN
PRINT "RA"	idle	XX	ATN,REN	
PRINT "RM1";Z\$	UNL primary listener idle	3F 31 XX	ATN ATN,REN ATN,REN	LSTN LSTN LSTN
PRINT "LL"	LLO idle	11 XX	ATN ATN	LSTN
PRINT "LA"	idle	XX	ATN,REN	
PRINT "LO1";Z\$	UNL primary listener GTL idle	3F 31 01 XX	ATN ATN ATN ATN	LSTN LSTN LSTN LSTN
INPUT "SR";G\$	idle	XX	[SRQ]	
PRINT "SPA";Z\$; INPUT H\$	SPE primary talker UNL MLA status byte SPD idle	18 41 3F 20 (40) 19 XX	SRQ,ATN SRQ,ATN SRQ,ATN SRQ,ATN [SRQ] [SRQ],ATN [SRQ],ATN	LSTN LSTN LSTN LSTN TALK LSTN LSTN

<u>COMMAND</u>	<u>DIO MNEMONIC</u>	<u>DIO (HEX)</u>	<u>MGMNT</u>	<u>HND SHK</u>
INPUT "PP";I\$	pp byte idle	(40) XX	ATN,EOI ATN	
PRINT "PE1A2B"	UNL primary listener PPC PPE + SPPP * primary listener PPC PPE + SPPP * idle	3F 31 05 61 32 05 62 XX	ATN ATN ATN ATN ATN ATN ATN ATN	LSTN LSTN LSTN LSTN LSTN LSTN LSTN LSTN
PRINT "PD1";Z\$	UNL primary listener PPC PPD idle	3F 31 05 70 XX	ATN ATN ATN ATN ATN	LSTN LSTN LSTN LSTN LSTN
PRINT "PU"	PPU idle	15 XX	ATN ATN	LSTN
PRINT "AB"	idle	XX	IFC	
PRINT "LF1"				
PRINT "WT1";Z\$;"DATA"	MTA UNL primary listener D A T A <CR> <LF> idle	40 3F 31 44 41 54 41 0D 0A XX	ATN ATN ATN     EOI ATN	LSTN LSTN LSTN LSTN LSTN LSTN LSTN LSTN LSTN LSTN

\* SPPP refers to the Sense bit and the 3 bit assignment bits.  
The code for PPE is logically ORed with these bits. Refer to  
the section on the PE command for more details.

<u>COMMAND</u>	<u>DIO</u> <u>MNEMONIC</u>	<u>DIO</u> <u>(HEX)</u>	<u>MGMNT</u>	<u>HNDSHK</u>
PRINT "ES!"				
PRINT "WT1";Z\$;"DATA!";	MTA	40	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	D	44		LSTN
	A	41		LSTN
	T	54		LSTN
	A	41		LSTN
	!	21	EOI	LSTN
	idle	XX	ATN	
 PRINT "SC1"	 idle	 XX	 ATN	
 PRINT "DV16"				
PRINT "WT1";Z\$;"DATA"	MTA	50	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	D	44		LSTN
	A	41		LSTN
	T	54		LSTN
	A	41		LSTN
	<CR>	0D	EOI	LSTN
	idle	XX	ATN	
 PRINT "UT"	 UNT	 5F	 ATN	 LSTN
	idle	XX	ATN	



# APPENDIX C

## 9914 REGISTERS

You will need to use the information in this appendix only if you want to send information directly to the registers contained in the TMS 9914 chip on the IEEE-488 card. Details on the use of the 9914 are beyond the scope of this document. Please refer to the 9914 specification sheets if you want or need more information.

RS2, RS1 and RS0 are the register address lines on the 9914. They are connected to the three low address lines on the Apple bus.

The N in the Apple addresses stands for eight plus the number of the slot into which the GPIB is inserted. For example, if the GPIB is installed in slot 3 then the address of the Auxiliary Command Register is C0B3 or C0BB. Because address bit 3 is not decoded, two addresses can access each of the 9914 registers.

<u>9914 REGISTER</u>			<u>APPLE ADDRESS</u>	
<u>RS2</u>	<u>RS1</u>	<u>RS0</u>	<u>READ</u>	
0	0	0	INT STATUS 0	C0N0 C0N8
0	0	1	INT STATUS 1	C0N1 C0N9
0	1	0	ADDRESS STATUS	C0N2 C0NA
0	1	1	BUS STATUS	C0N3 C0NB
1	0	0	ADDRESS SWITCH 1	C0N4 C0NC
1	1	0	CMD PASS THROUGH	C0N6 C0NE
1	1	1	DATA IN	C0N7 C0NF
			<u>WRITE</u>	
<u>RS2</u>	<u>RS1</u>	<u>RS0</u>		
0	0	0	INT MASK 0	C0N0 C0N8
0	0	1	INT MASK 1	C0N1 C0N9
0	1	1	AUXILIARY CMD	C0N3 C0NB
1	0	0	ADDRESS REG	C0N4 C0NC
1	0	1	SERIAL POLL	C0N5 C0ND
1	1	0	PARALLEL POLL	C0N6 C0NE
1	1	1	DATA OUT	C0N7 C0NF

NOTE: The 6502 performs a read operation before each write operation. Since some registers on the 9914 are reset by a read, you must take care that the proper action is taken. Refer to a 6502 Programming Manual and the 9914 specifications for further details.

An example of the use of some of the registers in the 9914 can be found in Appendix E.



## APPENDIX D

# PROGRAMMING AIDS

This appendix contains a series of helpful subroutines and programs for learning about and using the Apple IEEE-488 Interface Card.

### PROGRAM 1: CHR\$ FUNCTION FOR INTEGER BASIC

---

Purpose: This function gives you the ability to generate those characters that cannot be sent by the Apple's keyboard (see Tables A-3 and A-4 in Appendix A). It should be used with Integer BASIC in place of the CHR\$ function available to APPLESOFT.

Setup: Place the following lines of code somewhere in your program. Of course you may change the line numbers, but don't change anything else!! Line 120 POKes the proper ASCII code right between the " marks in the statement CHR\$="A".

Code:

```
110 CHS = CHR +128* (CHR<128)

120 LC1= PEEK (224): LC2= PEEK (
    225)-(LC1>243): POKE 79+LC1-
    256*(LC2>127)+(LC2-255*(LC1>
    127))*256,CHS:CHR$="A":RETURN
```

Use:

- 1) Place the ASCII code of the desired character into CHR.
- 2) GOSUB 110 (or the line number of your first line).
- 3) The value returned in CHR\$ is the character you want!

Errors: If line 120 is used in a long program on an Apple with more than 32K bytes of memory, there is a very remote chance that you will get a \*\*\* >32767 ERR. If this should happen, insert a REM statement with about 80 characters on a higher line number to force the CHR\$ function down in memory.

Source: Contributed Programs Volumes 3-5 (contains an explanation!)  
Apple Product #A2L0014

## PROGRAM 2: ONERR ROUTINE

---

Purpose: The ONERR..GOTO statement can be used to detect and recover from the BUS error. Since the BUS error cannot occur during normal bus protocol, it indicates that an abnormal situation exists; for example, it may occur if the bus is not connected properly.

Code:

```
5 DIM N$(5)
10 D$ = CHR$(4) :REM CTRL-D = DOS COMMAND
20 Z$ = CHR$(26) :REM CTRL-Z = BUS COMMAND
25 PRINT
30 ONERR GOTO 500 :REM ONERR ROUTINE AT 500
40 PRINT D$;"PR#3" :REM ENABLE OUTPUT TO BUS
50 PRINT D$;"IN#3" :REM ENABLE INPUT FROM BUS
55 PRINT "SC1" :REM SET SCREEN TO ON
60 PRINT "WT1"; CHR$(243); Z$;"ONERR GOTO TEST"
61 :REM DO A NORMAL WRITE TO LISTENER 1
62 :REM ERROR IF BUS NOT CONNECTED
70 PRINT D$;"PR#0" :REM ENABLE OUTPUT TO SCREEN
80 PRINT D$;"IN#0" :REM ENABLE INPUT FROM KEYBOARD
90 END
500 PRINT : PRINT :REM ONERR ROUTINE
510 PRINT "'ONERR GOTO' GOT US HERE"
511 PRINT : PRINT "ERROR FLAG = "; :REM SHOULD BE 49
512 PRINT PEEK(222) : PRINT :REM PRINT ERROR NUMBER
513 PRINT "ERROR OCCURRED AT LINE ";
514 PRINT PEEK(218) + 256 * PEEK(219)
515 :REM USE DOS ONERR LOCATIONS
516 :REM TO PRINT LINE NUMBER
520 POKE 216,0 :REM RESET ONERR FLAG
530 PRINT : PRINT
540 INPUT "FIX BUS THEN HIT RETURN TO CONTINUE";N$
550 GOTO 25
```

Use: To test the ONERR routine:

- 1) Enter the program.
- 2) Make sure the bus is disconnected.
- 3) Run the program.
- 4) Reconnect the bus and continue.

Note: A good discussion of ONERR can be found in the DOS 3.2 Manual, Apple Product #A2L0012.



## PROGRAM 3: SKIP REINITIALIZE FUNCTION

**Purpose:** When a PR#n and IN#n are issued with the IEEE-488 card in slot n, all the card's options are initialized to their default values. The subroutine beginning at line 20000 does a PR#3, IN#3 without reinitializing the card. It can be adapted to any slot.

### Code:

```
1000 DIM A$(10) :REM DELETE LINES 100-120
1100 DIM C$(10) :REM IF YOU ARE USING
1200 DIM G$(10) :REM APPLESOFT.
1300 Z$ = "" :REM Z$ CONTAINS CTRL-Z
1400 D$ = "" :REM D$ CONTAINS CTRL-D
10000 PRINT D$;"IN#3" :REM ENABLE INPUT FROM SLOT 3
10100 PRINT D$;"PR#3" :REM ENABLE OUTPUT TO SLOT 3
10200 PRINT "SC1" :REM TURN ON THE SCREEN
10300 PRINT "WT123";Z$;"HI" :REM SEND "HI" TO THE GPIB & SCREEN
10500 INPUT "SR";G$ :REM SERVICE REQUESTED?
10600 PRINT D$;"IN#0" :REM ENABLE INPUT FROM KEYBOARD
10700 PRINT D$;"PR#0" :REM ENABLE OUTPUT TO SCREEN ONLY
10800 INPUT "THIS LINE DOES NOT APPEAR ON THE BUS — PRESS RETURN TO
CONTINUE.";A$
10900 GOSUB 20000 :REM RESTORE INPUTS AND OUTPUTS
11000 PRINT "WC123";Z$;"5";Z$;"-DATA"; :REM SEND "-DATA" TO
:REM GPIB AND SCREEN
11100 PRINT "RDA";Z$; :REM SEND COMMAND, ADRS, CTRL-Z
11200 INPUT C$ :REM AND GET INPUT STRING
11300 PRINT D$;"IN#0" :REM ENABLE INPUT FROM KEYBOARD
11400 PRINT D$;"PR#0" :REM ENABLE OUTPUT TO SCREEN
12000 END
1998 REM ---- THIS SUBROUTINE DOES A PR#3/IN#3 WITHOUT RESETTNG
1999 REM ---- THE GPIB DEFAULT VALUES.
20000 POKE 54,6 :REM POKE ADDRESS OF CHARACTER
20100 POKE 55,195 :REM OUTPUT SUBROUTINE.
:REM 195 = 192 + SLOT NUMBER
20200 POKE 56,3 :REM POKE ADDRESS OF CHARACTER
20300 POKE 57,195 :REM INPUT SUBROUTINE.
:REM 195 = 192 + SLOT NUMBER
20400 CALL 10002 :REM CALL DOS INITIALIZATION
20500 RETURN
```

**Use:**

- 1) Type in the code.
- 2) If you don't have DOS, delete line 20400 and the D\$'s.
- 3) Run it.
- 4) You can use lines 20000-20500 as a general subroutine.

**Errors:** This code will not work with DOS 3.1.

**Notes:**

- 1) DOS input and output locations are discussed in the chapter on Input and Output in the DOS 3.2 (or 3.3) Manual.
- 2) Lines 100-120 should only be used with Integer BASIC.

## PROGRAM 4: INPUT WITH COMMAS

---

Purpose: APPLESOFT interprets a comma in a data string as the end of input. Use this routine to input data from a device that sends commas. Input terminates when a CR is read. (Note that you can easily modify it to terminate on EOS as well.)

Setup: Type in the lines of code

Code:

```
5 REM CAN ONLY BE USED IN APPLESOFT
6 REM INPUT STRING IS STORED IN B$
7 REM INPUT IS TERMINATED WITH CR
25 A$ = "" : B$ = "" :REM STRINGS INITIALLY EMPTY
30 GET A$ :REM READ A CHARACTER AND
40 PRINT A$ :REM ECHO IT. SEE NOTE.
50 IF ASC (A$) = 13 THEN RETURN:REM RETURN ON CR
60 B$ = B$ + A$ :REM ADD CHARACTER TO STRING
70 GOTO 30
```

Use: 1) Call with GOSUB 25 (or whatever you call that line)  
2) Input string will be returned in B\$

Notes: 1) The firmware expects strings to be read from the bus using the INPUT statement. Since the INPUT statement echoes characters to the screen, a GET statement must be followed by a PRINT statement to simulate this echo. In this way you can fool the firmware into thinking that normal input is being done.

Examples: This routine is used in the second sample program in Appendix G. Refer to that example to see how it can be used.

## APPENDIX E

# FAST TRANSFER ROUTINES

**Purpose:** These routines are an example of how one can do fast block transfers over the GPIB. These routines communicate directly with the 9914, as opposed to normal GPIB transfers, which send a string to DOS, which uses the Apple Monitor Input or Output Registers to transfer the string character by character to the IEEE-488 card, which finally sends it to the 9914. These routines are also an example of how a machine language subroutine can be used to implement one of the GPIB instructions. This will be useful if you want to use the GPIB with the Pascal System. See Appendix F for more details on how to use the GPIB with Pascal.

**Background:** These routines were generated using the Apple 6502 Assembler/Editor. The code is ORG'd at \$5000 and the data at \$5200. You will probably need to change these locations depending on your application. The code assumes that your card is in slot 3, and that your EOS character is CR. (See the declarations of EOS and SLOT at \$5200 and \$5202.) The end of this Appendix explains how these routines can be run and tested.

## GPIB SUBR. DEMO

This BASIC test program may be used to call the machine code subroutines. It first POKes an RTS in place of the BRK at location 21569 (\$5441); the BRK is used for test purposes. It then calls the test routine at \$5400, which in turn calls the other routines. Finally, it executes some GPIB instructions to show that the routines don't interfere with normal GPIB operation.

```
10 DIM S$(30)           : REM  DELETE THIS LINE FOR APPLESOFT
15 POKE 21569,96         : REM  PUT AN RTS IN PLACE OF BRK
20 CALL 21504            : REM  $5400 IS THE TEST PROGRAM
30 Z$ = ""              : REM  Z$ IS CTRL-Z, GPIB COMMAND
40 D$ = ""              : REM  D$ IS CTRL-D, DOS COMMAND
50 PRINT D$;"PR#3"      : REM  ENABLE OUTPUT TO CARD
60 PRINT D$;"IN#3"      : REM  ENABLE INPUT FROM CARD
65 PRINT "SC1"          : REM  TURN SCREEN ON
70 PRINT "RDB";Z$;      : REM  READ FROM TALKER "B"
75 INPUT S$
80 PRINT D$;"PR#0"      : REM  ENABLE OUTPUT TO SCREEN
90 PRINT D$;"IN#0"      : REM  ENABLE INPUT FROM KEYBOARD
100 PRINT "S$ = ";S$    : REM  ECHO INPUT STRING
110 PRINT D$;"PR#3"     : REM  ENABLE OUTPUT TO CARD
120 PRINT D$;"IN#3"     : REM  ENABLE INPUT FROM CARD
125 PRINT "SC1"         : REM  TURN SCREEN ON
130 PRINT "WT1";Z$;"FIRMWARE" : REM  SEND "FIRMWARE" TO "1"
140 PRINT D$;"PR#0"     : REM  ENABLE OUTPUT TO SCREEN
150 PRINT D$;"IN#0"     : REM  ENABLE INPUT FROM KEYBOARD
200 END
```

# SUBROUTINES

All of the following routines have arbitrary starting addresses. Relocate them according to your system configuration. You must also relocate the seven bytes of storage. The location of the buffers for data and device lists is up to you.

## GENERAL CONTROL VALUES

The general control values assume that the Apple is device #0.

C080:	19	GPIB	EQU	\$C080	;SET BASE ADDRESS OF ROM
0000:	20	;			OFFSET FOR SLOT 3 ASSUMED
C080:	21	INT0	EQU	GPIB+0	;INTERRUPT REG. 0
C080:	22	INTM0	EQU	GPIB+0	;INTRP MASK REG 0
0020:	23	BIM	EQU	\$20	;BYTE IN MASK
0010:	24	BOM	EQU	\$10	;BYTE OUT MASK
0008:	25	EOIMK	EQU	\$08	;EOI MASK
0000:	26	;			
C081:	27	INTM1	EQU	GPIB+1	; INTRP MASK REG 1
0000:	28	;			
C083:	29	AUXCMD	EQU	GPIB+3	;AUXILIARY COMMAND REGISTER
0080:	30	RESET	EQU	\$80	;SOFTWARE CHIP RESET
0000:	31	RSTCLR	EQU	\$00	;STOP SOFTWARE RESET
0083:	32	HDFA	EQU	\$83	;HOLDOFF ON ALL DATA
0003:	33	HDACLR	EQU	\$03	;CLEAR HOLDOFF ON ALL
0089:	34	LON	EQU	\$89	;LISTEN ONLY
008A:	35	TON	EQU	\$8A	;TALK ONLY
008F:	36	SIC	EQU	\$8F	;SEND INTERFACE CLEAR
000F:	37	SICLR	EQU	\$0F	;CLEAR SIC
0002:	38	RHDF	EQU	\$02	;RELEASE RFD HOLDOFF
0008:	39	FEOI	EQU	\$08	;SEND EOI WITH NEXT BYTE
000B:	40	GTS	EQU	\$0B	;GO TO STANDBY
000C:	41	TCA	EQU	\$0C	;TAKE CONTROL ASYNCHRONOUSLY
000D:	42	TCS	EQU	\$0D	;TAKE CONTROL SYNCHRONOUSLY
0000:	43	;			
C087:	44	DIN	EQU	GPIB+7	;DATA IN REGISTER
C087:	45	DOUT	EQU	GPIB+7	;DATA OUT REGISTER
0000:	46	;			
0000:	47	;			
0000:	47	;			
0000:	48	;			
0020:	49	MLA	EQU	\$20	;MY LISTEN ADDRESS
0040:	50	MTA	EQU	\$40	;MY TALK ADDRESS
003F:	51	UNL	EQU	\$3F	;UNIVERSAL UNLISTEN

## INITIALIZE 9914

All of the routines make common assumptions about the state of the 9914 chip upon entry, and all will leave the 9914 in this same state upon exit. The INIT routine sets the 9914 to this desired state. These assumptions are:

- BO set
- All interrupts masked off
- TON (talk only) on
- Not listener active (LA)
- No holdoffs enabled or active
- No 6502 registers destroyed
- Status obtained by polling the 9914

SUBROUTINES CALLED: WAIT

```

5000:          101      ORG  $5000
5000:          102 ;
5000:08        103 INIT:  PHP          ;SAVE REGISTERS
5001:48        104      PHA
5002:98        105      TYA
5003:48        106      PHA
5004:8A        107      TXA
5005:48        108      PHA
5006:AC 02 52  109      LDY  SLOT      ;GPB SLOT INDICATOR
5009:A9 80     110      LDA  #RESET    ;RESET 9914
500B:99 83 C0  111      STA  AUXCMD,Y
500E:A9 00     112      LDA  #RSTCLR   ;TURN OFF RESET
5010:99 80 C0  113      STA  INTM0,Y    ;CLEAR INTERRUPTS
5013:99 81 C0  114      STA  INTM1,Y    ;CLEAR INTERRUPTS
5016:99 83 C0  115      STA  AUXCMD,Y
5019:A9 8F     116      LDA  #SIC      ;SEND INTERFACE CLEAR
501B:99 83 C0  117      STA  AUXCMD,Y  ;AND TAKE CONTROL
501E:A2 04     118      LDX  #4        ;5.12 MSEC DELAY
5020:A0 00     119      LDY  #0        ;INITIALIZE INNER LOOP
5022:88        120 INIT1: DEY          ;5 USEC INNER LOOP
5023:D0 FD     121      BNE  INIT1      ;1280 USEC TOTAL
5025:CA        122      DEX
5026:D0 FA     123      BNE  INIT1
5028:AC 02 52  124      LDY  SLOT
502B:A9 0F     125      LDA  #SICLR    ;CLEAR INTERFACE CLR
502D:99 83 C0  126      STA  AUXCMD,Y
5030:A9 8A     127      LDA  #TON      ;SET TALK ONLY MODE
5032:99 83 C0  128      STA  AUXCMD,Y
5035:20 32 51  129      JSR  WAIT      ;WAIT FOR BYTE OUT
5038:          130 ;                TO INDICATE CONTROL TAKEN
5038:68        131      PLA          ;RESTORE REGISTERS
5039:AA        132      TAX
503A:68        133      PLA
503B:A8        134      TAY
503C:68        135      PLA
503D:28        136      PLP
503E:60        137      RTS          ;RETURN TO CALLING POINT

```

## SEND ROUTINE

The send routine can be used in place of the WT or WC command. It terminates either when COUNT bytes have been sent or when EOS is sent. It asserts EOI in both cases.

INPUTS: ABUF = Pointer to the listener list  
 DBUF = Pointer to the data buffer  
 COUNT = Number of bytes to send; 0 causes no data.  
 EOS = The terminating character

OUTPUTS: None to software

SUBROUTINES CALLED: WAIT

REGISTERS DESTROYED: None

503F:08	157	SEND:	PHP	SAVE REGISTERS
5040:48	158		PHA	
5041:98	159		TYA	
5042:48	160		PHA	
5043:8A	161		TXA	
5044:48	162		PHA	
5045:AC 02 52	163		LDY SLOT	; GPIB SLOT INDICATOR
5048:A9 40	164		LDA #MTA	; MY TALK ADDRESS
504A:99 87 C0	165		STA DOUT,Y	
504D:20 32 51	166		JSR WAIT	; WAIT FOR BYTE OUT
5050:A9 3F	167		LDA #UNL	; UNIVERSAL UNLISTEN
5052:99 87 C0	168		STA DOUT,Y	
5055:A2 00	169		LDX #0	; BUFFER INDEX
5057:BD 03 52	170	SEND1:	LDA ABUF,X	; GET LISTENER ADDRESS
505A:C9 20	171		CMP #\$20	; LOWER LIMIT OF RANGE
505C:30 10	172		BMI SEND2	
505E:C9 3F	173		CMP #\$3F	; UPPER LIMIT
5060:10 0C	174		BPL SEND2	
5062:20 32 51	175		JSR WAIT	; WAIT FOR UNL BYTE OUT
5065:BD 03 52	176		LDA ABUF,X	; GET LISTENER AGAIN
5068:99 87 C0	177		STA DOUT,Y	; OUTPUT IT TO GPIB
506B:E8	178		INX	
506C:10 E9	179		BPL SEND1	; ALWAYS TAKEN
506E:	180	;		(MAX OF 31 LISTENERS!)
506E:A9 0B	181	SEND2:	LDA #GTS	; GO TO STANDBY
5070:99 83 C0	182		STA AUXCMD,Y	
5073:20 32 51	183		JSR WAIT	; WAIT FOR ATN TO BE
5076:	184	;		DE-ASSERTED
5076:A2 00	185		LDX #0	; INITIALIZE DATA BUF INDEX
5078:EC 01 52	186		CPX COUNT	; IS INITIAL COUNT = 0?
507B:F0 28	187		BEQ SEND6	; YES--GO QUIT
507D:CE 01 52	188	SEND3:	DEC COUNT	
5080:F0 11	189		BEQ SEND5	; IF COUNT FINISHED GO SEND EOI
5082:	190	;		WITH LAST CHARACTER
5082:BD 05 52	191		LDA DBUF,X	; GET DATA BYTE
5085:CD 00 52	192		CMP EOS	; IS IT LAST CHARACTER?
5088:F0 09	193		BEQ SEND5	; YES--GO HANDLE IT
508A:99 87 C0	194		STA DOUT,Y	; OUTPUT DATA TO GPIB
508D:20 32 51	195	SEND4:	JSR WAIT	; WAIT FOR DATA BYTE OUT
5090:E8	196		INX	
5091:D0 EA	197		BNE SEND3	; ALWAYS TAKEN

5093:A9 08	198	SEND5:	LDA	#FEOI	;FORCE EOI W/ EOS
5095:99 83 C0	199		STA	AUXCMD,Y	
5098:BD 05 52	200		LDA	DBUF,X	;GET BYTE AGAIN
509B:99 87 C0	201		STA	DOUT,Y	;OUTPUT TO GPIB
509E:E8	202		INX		
509F:CE 01 52	203		DEC	COUNT	;FOR CONSISTENCY
50A2:20 32 51	204		JSR	WAIT	;WAIT FOR LAST BYTE OUT
50A5:A9 0C	205	SEND6:	LDA	#TCA	;TAKE CONTROL
50A7:99 83 C0	206		STA	AUXCMD,Y	
50AA:20 32 51	207		JSR	WAIT	;WAIT FOR CONTROL
50AD:68	208		PLA		;RESTORE REGISTERS
50AE:AA	209		TAX		
50AF:68	210		PLA		
50B0:A8	211		TAY		
50B1:68	212		PLA		
50B2:28	213		PLP		
50B3:00	214		RTS		;RETURN TO CALL POINT

## RECEIVE ROUTINE

The receive routine can be used in place of the RD or RC command. It terminates either when COUNT bytes have been read or when EOS is read or when EOI goes high.

INPUTS: ABUF = Pointer to the talker list  
 DBUF = Pointer to the data buffer  
 COUNT = Number of bytes to receive; 0 receives no data  
 EOS = The terminating character

OUTPUTS: Data buffer is filled

SUBROUTINES CALLED: WAIT

REGISTERS DESTROYED: None

50B4:08	235	RECV:	PHP		;SAVE REGISTERS
50B5:48	236		PHA		
50B6:98	237		TYA		
50B7:48	238		PHA		
50B8:8A	239		TXA		
50B9:48	240		PHA		
50BA:AC 02 52	241		LDY	SLOT	;GPIB SLOT IDENTIFIER
50BD:A2 00	242		LDX	#0	
50BF:AD 03 52	243		LDA	ABUF	;GET TALKER DEVICE NO.
50C2:99 87 C0	244		STA	DOUT,Y	;OUTPUT IT TO GPIB
50C5:20 32 51	245		JSR	WAIT	;WAIT FOR BYTE OUT
50C8:A9 3F	246		LDA	#UNL	;UNIVERSAL UNLISTEN
50CA:99 87 C0	247		STA	DOUT,Y	;OUTPUT IT TO GPIB
50CD:20 32 51	248		JSR	WAIT	;WAIT FOR BYTE OUT
50D0:A9 20	249		LDA	#MLA	;MY LISTEN ADDRESS
50D2:99 87 C0	250		STA	DOUT,Y	;OUTPUT IT TO GPIB
50D5:A9 83	251		LDA	#HDFA	;ENABLE HOLDOFF ON ALL
50D7:99 83 C0	252		STA	AUXCMD,Y	
50DA:A9 89	253		LDA	#LON	;LISTEN ONLY-CLEAR TON
50DC:99 83 C0	254		STA	AUXCMD,Y	
50DF:A9 0B	255		LDA	#GTS	;GO TO STANDBY
50E1:99 83 C0	256		STA	AUXCMD,Y	
50E4:20 32 51	257		JSR	WAIT	;WAIT FOR MLA BYTE OUT



50E7:B9 80 C0	258	RECV1:	LDA	INT0,Y	;WAIT FOR BYTE IN
50EA:29 28	259		AND	#EOIMK+BIM	;TEST FOR EOI OR BI
50EC:F0 F9	260		BEQ	RECV1	
50EE:29 08	261		AND	#EOIMK	;IS IT EOI?
50F0:D0 18	262		BNE	RECV2	;YES
50F2:B9 87 C0	263		LDA	DIN,Y	;GET DATA BYTE FROM GPIB
50F5:9D 05 52	264		STA	DBUF,X	;STORE IT IN BUFFER
50F8:CD 00 52	265		CMP	EOS	;IS IT THE EOS CHAR?
50FB:F0 14	266		BEQ	RECV3	;YES
50FD:E8	267		INX		
50FE:A9 02	268		LDA	#RHDF	;RELEASE HOLDOFF
5100:99 83 C0	269		STA	AUXCMD,Y	
5103:CE 01 52	270		DEC	COUNT	
5106:F0 0C	271		BEQ	RECV4	;GO IF COUNT =0
5108:D0 DD	272		BNE	RECV1	;ALWAYS TAKEN--GO FOR MORE
510A:B9 87 C0	273	RECV2:	LDA	DIN,Y	;GET LAST BYTE
510D:9D 05 52	274		STA	DBUF,X	
5110:E8	275		INX		
5111:CE 01 52	276	RECV3:	DEC	COUNT	
5114:A9 0D	277	RECV4:	LDA	#TCS	;TAKE CONTROL SYNC
5116:99 83 C0	278		STA	AUXCMD,Y	
5119:20 32 51	279		JSR	WAIT	;WAIT FOR CONTROL
511C:A9 02	280		LDA	#RHDF	;RELEASE HOLDOFF
511E:99 83 C0	281		STA	AUXCMD,Y	
5121:A9 8A	282		LDA	#TON	;SET TALK ONLY, CLR LON
5123:99 83 C0	283		STA	AUXCMD,Y	
5126:A9 03	284		LDA	#HDA CLR	;CLEAR HOLDOFF MODE
5128:99 83 C0	285		STA	AUXCMD,Y	
512B:68	286		PLA		;RESTORE REGISTERS
512C:AA	287		TAX		
512D:68	288		PLA		
512E:A8	289		TAY		
512F:68	290		PLA		
5130:28	291		PLP		
5131:60	292		RTS		;RETURN TO CALL POINT

## WAIT ROUTINE

The wait routine reads the INT0 register until the Byte Out or Byte In bit is set.

INPUTS: Register Y contains the slot number

OUTPUTS: None

PROCEDURES CALLED: None

REGISTERS DESTROYED: None

5132:B9 80 C0	310	WAIT:	LDA	INT0,Y	;GET BYTE OUT FLAG
5135:29 10	311		AND	#BOM	
5137:F0 F9	312		BEQ	WAIT	;LOOP UNTIL SET
5139:60	313		RTS		



## DATA BUFFERS

The following data buffers work with the test routine given. You may have to change them to suit your purposes. Notice that the default value of EOS is CR, the default number of bytes counted is 255, and the default slot is 3. ABUF, the list of listeners or talkers is only large enough to contain one listener or talker; this may be set according to your needs. This list may actually end with any non-listener or non-talker depending on which routine is using it.

```
5200:          325          ORG  $5200
5200:          326 ;
5200:0D        327 EOS      DFB  $0D          ;EOS CHARACTER--DEFAULT IS CR
5201:FF        328 COUNT   DFB  $FF          ;BYTE COUNT FOR DATA BUFFER
5202:          329 ;                      --DEFAULT IS 255
5202:30        330 SLOT    DFB  $30          ;OFFSET TO SLOT 3 ADDRESSES
5203:          331 ;                      --DEFAULT IS SLOT #3
5203:          332 ABUF     DS   2            ;DEVICE LIST
5205:          333 ;                      --MUST END WITH $FF BYTE
5205:          334 DBUF     DS   256          ;DATA BUFFER
```

## SUBROUTINE TEST

This routine tests the subroutines that have already been given. It is called by the GPIB SUBR DEMO given at the beginning of this section. It sends data to listener \$21 (ASCII "!") and receives data from talker \$41 (ASCII "A"). The data buffer (DBUF) is filled with the numbers 1 through 5 followed by a CR. This data is first sent with COUNT as initialized (255) so that the CR terminates output. Then it is sent with a COUNT of 5; thus the CR doesn't get sent. The BRK in line 376 was replaced by an RTS instruction, thus the REC V routine is only called once, and the loop is never executed.

Notes: The code assumes that <EOS> is set to <CR>. If you have changed your EOS, the first call to SEND will send 255 bytes.

```
5400:          347          ORG  $5400
5400:          348 ;
5400:20 00 50    349          JSR  INIT          ;SET UP GPIB INITIALLY
5403:EA          350          NOP
5404:A9 21      351          LDA  #$21          ;LISTENER ADDRESS
5406:8D 03 52    352          STA  ABUF
5409:A9 FF      353          LDA  #$FF          ;LIST END
540B:8D 04 52    354          STA  ABUF+1
540E:A9 01      355          LDA  #1
5410:8D 05 52    356          STA  DBUF          ;DATA TO BE SENT
5413:A9 02      357          LDA  #2
5415:8D 06 52    358          STA  DBUF+1
5418:A9 03      359          LDA  #3
541A:8D 07 52    360          STA  DBUF+2
541D:A9 04      361          LDA  #4
541F:8D 08 52    362          STA  DBUF+3
5422:A9 05      363          LDA  #5
5424:8D 09 52    364          STA  DBUF+4
```

5427:A9 0D	365	LDA #S0D	;6 BYTES OF DATA INCL CR
5429:8D 0A 52	366	STA DBUF+5	
542C:20 3F 50	367	JSR SEND	
542F:EA	368	NOP	
5430:A9 05	369	LDA #5	;5 BYTES OF DATA, NO CR
5432:8D 01 52	370	STA COUNT	
5435:20 3F 50	371	JSR SEND	
5438:EA	372	NOP	
5439:A9 41	373	LDA #S41	;TALKER ADDRESS
543B:8D 03 52	374	STA ABUF	
543E:20 B4 50	375	JSR RECVR	
5441:00	376	BRK	
5442:4C 3E 54	377	JMP LOOP	
5445:	378 ;		
5445:	379 ;END		

## SYMBOL TABLE

5203 ABUF	C083 AUXCMD	20 BIM	10 BOM
5201 COUNT	5205 DBUF	C087 DIN	C087 DOUT
08 EOIMK	5200 EOS	08 FEOI	C080 GPIB
0B GTS	03 HDACLR	83 HDFA	5000 INIT
5022 INIT1	C080 INT0	C080 INTM0	C081 INTM1
89 LON	543E LOOP	20 MLA	40 MTA
510A RECVR	50B4 RECVR	50E7 RECVR	5111 RECVR
5114 RECVR	80 RESET	02 RHDF	00 RSTCLR
5057 SEND1	506E SEND2	503F SEND	507D SEND3
?508D SEND4	5093 SEND5	50A5 SEND6	0F SICLR
8F SIC	5202 SLOT	0C TCA	0D TCS
8A TON	3F UNL	5132 WAIT	

## DATA RATES

When the GPIB does a WRITE operation, it transfers the data at a rate of 1.5 kilobytes per second. This does not include the roughly 3.5 millisecond interval during which processing of the instruction takes place. For a READ operation, the GPIB transfers data at a rate of about 1 kilobyte per second.

The performance of the IEEE-488 bus can be improved with the preceding routines. These routines, as listed, send and receive data at a rate of 20 kilobytes per second. Even this is not optimal. If the above routines are streamlined to transfer data by count only, they will send data at a rate of 50 kilobytes per second.

# RUNNING AND TESTING

Our version of the APPLESOFT program used to run these subroutines was called GPIB SUBR DEMO. The assembly language routine source code was saved in the file GPIB SUBR. The 6502 Assembler places the assembled code into the files:

```
GPIB SUBR.OBJ0 (routines)
GPIB SUBR.OBJ1 (data)
GPIB SUBR.OBJ2 (test code)
```

The routines may be tested with the following sequence of commands:

```
] BLOAD GPIB SUBR.OBJ0
] BLOAD GPIB SUBR.OBJ1
] BLOAD GPIB SUBR.OBJ2
] RUN GPIB SUBR DEMO
```

We recommend that you use a ZT 488 GPIB Analyzer to verify correct operation of the routines. The sequence of bus signals should be:

<u>DIO</u>	<u>MGMNT</u>	<u>DIO</u>	<u>MGMNT</u>	<u>DIO</u>	<u>MGMNT</u>	<u>DIO</u>	<u>MGMNT</u>	<u>DIO</u>	<u>MGMNT</u>
40	ATN	40	ATN	41	ATN	42	ATN	40	ATN
3F	ATN	3F	ATN	3F	ATN	3F	ATN	3F	ATN
21	ATN	21	ATN	20	ATN	20	ATN	31	ATN
1		1			INPUT		INPUT	46	
2		2			any		any	49	
3		3			sequence		sequence	52	
4		4			terminated		terminated	4D	
5		5	EOI		with		with	57	
0D	EOI				0D or EOI		0D or EOI	41	
								52	
								45	
								0D	EOI



# APPENDIX F

## PROGRAMMING IN PASCAL

Because the GPIB firmware is designed to interact with BASIC, the Monitor and DOS, it will not behave properly if commands are sent to the GPIB using the Pascal WRITE and Writeln commands. This does not, however, mean that the Apple GPIB cannot be used with the Pascal System.

If you are familiar with 6502 Assembly Language, and with the Apple Pascal Operating System, it is possible for you to implement most or all of the GPIB commands for use with the Pascal System.

### WHAT YOU MUST DO

---

What you must do is write a Pascal procedure for each of the GPIB instructions that you want to implement. Some of these procedures, for example SC, can be written entirely in Pascal. Others, however, must be at least partially written in 6502 assembly code. In the 6502 code you send commands directly to the 9914 chip.

Before you can even attempt to write 6502 routines, you must know what commands are required by the 9914. Obtain the TMS 9914 (made by Texas Instruments) specifications to find out what values must be sent to which of the registers in the 9914 to implement the GPIB instructions that you will need.

Look in Appendix C to determine the Apple addresses of the registers on the 9914. These are the addresses with which your assembly language routines must communicate. The routines that you write must communicate with the 9914 registers in the same way as the Send, Receive and Initialize routines given in Appendix E. (See the use of locations INT0, INTM0, INTM1, DIN and DOUT.) In fact, you can even use slightly modified versions of these three routines as the start of your Pascal GPIB package.

Now write a 6502 procedure that will carry out the function that you want to implement. All that you must do is link this procedure into the Pascal system.

Information on the Apple Pascal's 6502 Assembler directives may be found in the chapter on the 6502 Assembler in the Apple Pascal Operating System Reference Manual (Apple Product #A2L0028). This chapter explains how a procedure must be structured to be linked into the Pascal System.

The chapter on the Linker in the same manual explains the actual process of linking a procedure into the Pascal System. Finally, the Apple Pascal Language Reference Manual (Apple Product #A2L0027) tells how EXTERNAL procedures (all assembly language procedures are external to the host program) are declared and called from within a Pascal host program.



Since your assembly language routines will be communicating with addresses that are external to the Apple, it is very advisable that you use a logic analyzer such as the ZT488 GPIB ANALYZER to monitor the signals on the bus. This may be the only way that you can verify correct bus operation.

## APPENDIX G

# SAMPLE PROGRAMS

This appendix contains two sample programs for the IEEE-488 card. You won't be able to use these programs unless you happen to have an HP 3438A Multimeter or an HP 3586C Selective Level Meter.

Nevertheless, they serve as excellent examples of the types of programs that can be written for the GPIB.

## READ ROUTINE FOR HP 3438A

The HP3438A Multimeter has a device number of 3. Its listener address is "#" and its talk address is "C". The following program assumes that the IEEE-488 card is in slot 3. The data sent by the 3438A takes the form SD.DDDESDF<CR><LF> for which

S is the sign, either "+" or "-"  
D is a decimal digit from "0" to "9"  
E is an "E" indicating the exponent  
, is a delimiter between the value and the function code  
F is the function code, a digit between "1" and "5"  
<CR><LF> is carriage return followed by line feed

```
10 DIM S$(30): REM ----- DELETE FOR APPLESOFT
20 DIM C$(30): REM ----- DELETE FOR APPLESOFT
30 Z$ = "": REM      CTL-Z
40 D$ = "": REM      CTL-D
50 PRINT D$;"PR#3"
60 PRINT D$;"IN#3"
70 PRINT "SCI": REM ---- TURN THE SCREEN ON
80 PRINT "TG#";Z$: REM --- TRIGGER A READING
90 PRINT "LF1": REM ---- 3438A SENDS A <CR><LF>
100 PRINT "RDC";Z$;: REM - READ FROM TALKER "C"
110 INPUT S$,C$: REM ---- S$ GETS CHARACTERS BEFORE
120 REM ----- THE "," SENT BY THE 3438A
130 REM ----- C$ GETS CHARACTERS AFTER THE ","
140 PRINT D$;"PR#0": REM - SEND OUTPUT TO SCREEN
150 PRINT D$;"IN#0": REM - GET INPUT FROM KEYBOARD
160 PRINT "READING = ";S$;
170 REM ----- NEXT LINE COMPUTES VALUE OF FUNCTION
180 REM ----- CODE RETURNED BY THE 3438A
190 ON ASC ( LEFT$ ( C$,1)) - 48 GOTO 200,210,220,230,240
200 PRINT " DCV": GOTO 250
210 PRINT " ACV": GOTO 250
220 PRINT " DCI": GOTO 250
230 PRINT " ACI": GOTO 250
240 PRINT " OHMS": GOTO 250
250 INPUT "WOULD YOU LIKE ANOTHER READING(Y OR N) ?";A$
260 IF A$ = "Y" THEN GOTO 50
270 END
```

# GENERAL PURPOSE HP 3586C ROUTINE

---

```
10 REM      ROUTINE FOR HP3586C SELECTIVE LEVEL METER
20 REM      METER IS DEVICE 16
30 REM      LISTEN ADDRESS IS ASCII 30 = "0"
40 REM      TALK ADDRESS IS ASCII 50 = "P"
50 REM      ASSUMES CARD IN SLOT #3
60 D$ = "": REM      CONTROL-D
70 Z$ = "": REM      CONTROL-Z
80 DIM C$(100): REM  DELETE FOR APPLESOFT
90 GOSUB 590: REM  INITIALIZE CARD FOR I/O
100 GOSUB 660: REM  INITIALIZE APPLE FOR I/O
110 HOME
120 VTAB 5: HTAB 18: PRINT "MENU": PRINT : PRINT
130 HTAB 8: PRINT "1 : PRINT TO THE DEVICE"
140 HTAB 8: PRINT "2 : LISTEN TO DEVICE"
150 HTAB 8: PRINT "3 : CLEAR DEVICE"
160 HTAB 8: PRINT "4 : REMOTE ENABLE DEVICE"
170 HTAB 8: PRINT "5 : LOCAL LOCKOUT DEVICE"
180 HTAB 8: PRINT "6 : SET DEVICE TO LOCAL MODE"
190 HTAB 8: PRINT "7 : ABORT SEQUENCE"
200 HTAB 8: PRINT "8 : SEND LITERAL STRING"
210 HTAB 8: PRINT "9 : QUIT": PRINT : PRINT
220 HTAB 8: PRINT "WHAT IS YOUR COMMAND?"
230 GET A
240 ON A GOTO 300,350
250 GOSUB 690
260 ON A - 2 GOTO 480,490,500,510,520
270 GOSUB 660
280 ON A - 7 GOTO 530,580
290 GOTO 110: REM  NO SUCH COMMAND
300 PRINT "ENTER COMMAND AND DATA": REM  PRINT TO THE DEVICE
310 INPUT C$
320 GOSUB 690: REM  INITIALIZE CARD FOR I/O
330 PRINT "WT0";Z$;C$: REM  SEND COMMAND TO METER
340 GOTO 100: REM  GET NEXT COMMAND
350 PRINT "INCOMING DATA IS..."
360 GOSUB 690: REM  INITIALIZE CARD FOR I/O
370 PRINT "TG0";Z$; REM  DEVICE MUST BE TRIGGERED BEFORE READING
380 PRINT "RDP";Z$; REM  READ FROM DEVICE
390 C$ = "": REM  SET C$ TO EMPTY
400 GET AS: PRINT AS;: REM  GET CHARACTER.
410 REM  --- SEE NOTE FOR REINITIALIZE FUNCTION.
420 IF ASC (AS) = 13 THEN 440: REM  REPEAT UNTIL <CR>
430 C$ = C$ + AS: GOTO 400: REM  SAVE CHARACTERS IN C$
440 GOSUB 660: REM  INITIALIZE APPLE FOR I/O
```



```

450 PRINT C$: REM PRINT INPUT STRING
460 PRINT "HIT ANY KEY TO CONTINUE..."
470 GET A$: GOTO 110: REM GET NEW COMMAND
480 PRINT "CA": GOTO 100: REM CLEAR ALL REMOTE DEVICES
490 PRINT "RA": GOTO 100: REM ENABLE DEVICES FOR REMOTE OPERATION
500 PRINT "LL": GOTO 100: REM LOCAL LOCKOUT DEVICE
510 PRINT "LA": GOTO 100: REM LOCKOUT DEVICE'S FRONT PANEL
520 PRINT "AB": GOTO 100: REM SET ALL DEVICES TO LOCAL MODE
530 PRINT "ENTER THE LINE TO BE TRANSMITTED:"
540 INPUT C$: REM READ LINE
550 GOSUB 690: REM INITIALIZE CARD FOR I/O
560 PRINT C$: REM TRANSMIT LINE
570 GOTO 100
580 END
590 PRINT D$;"PR#3": REM SEND OUTPUT TO IEEE-488 CARD
600 PRINT D$;"IN#3": REM GET INPUT FROM IEEE-488 CARD
610 PRINT "SC1": REM SET THE SCREEN TO ON
620 PRINT "RA": REM ENABLE ALL DEVICES FOR REMOTE OPERATION
630 PRINT "DV1": REM SET THE APPLE TO DEVICE 1
640 PRINT "LF1": REM SEND AND RECEIVE <LF> AFTER <CR>
650 RETURN
660 PRINT D$;"PR#0": REM SEND OUTPUT TO THE SCREEN
670 PRINT D$;"IN#0": REM GET INPUT FROM THE KEYBOARD
680 RETURN
690 PRINT
700 PRINT D$;"PR#3": REM SEND OUTPUT TO IEEE-488 CARD
710 PRINT D$;"IN#3": REM GET INPUT FROM IEEE-488 CARD
720 REM -AT THIS POINT ALL DEFAULT COMMANDS MUST BE RESET
730 REM -FOR EXAMPLE: LF1,SC1,DV1,ETC
740 RETURN

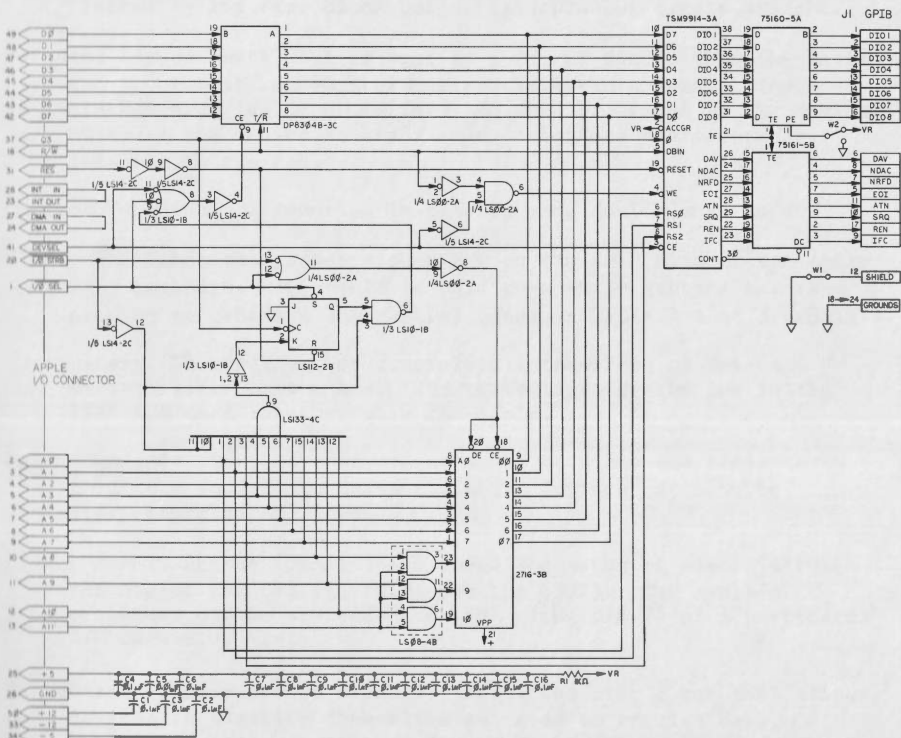
```

Notes: 1) Lines 410-450 contain the routine Input With Commas.  
Refer to Appendix D for a more complete explanation of this code.

# GENERAL PURPOSE INTERFACE CARD

# APPENDIX H

## SCHEMATIC DIAGRAM



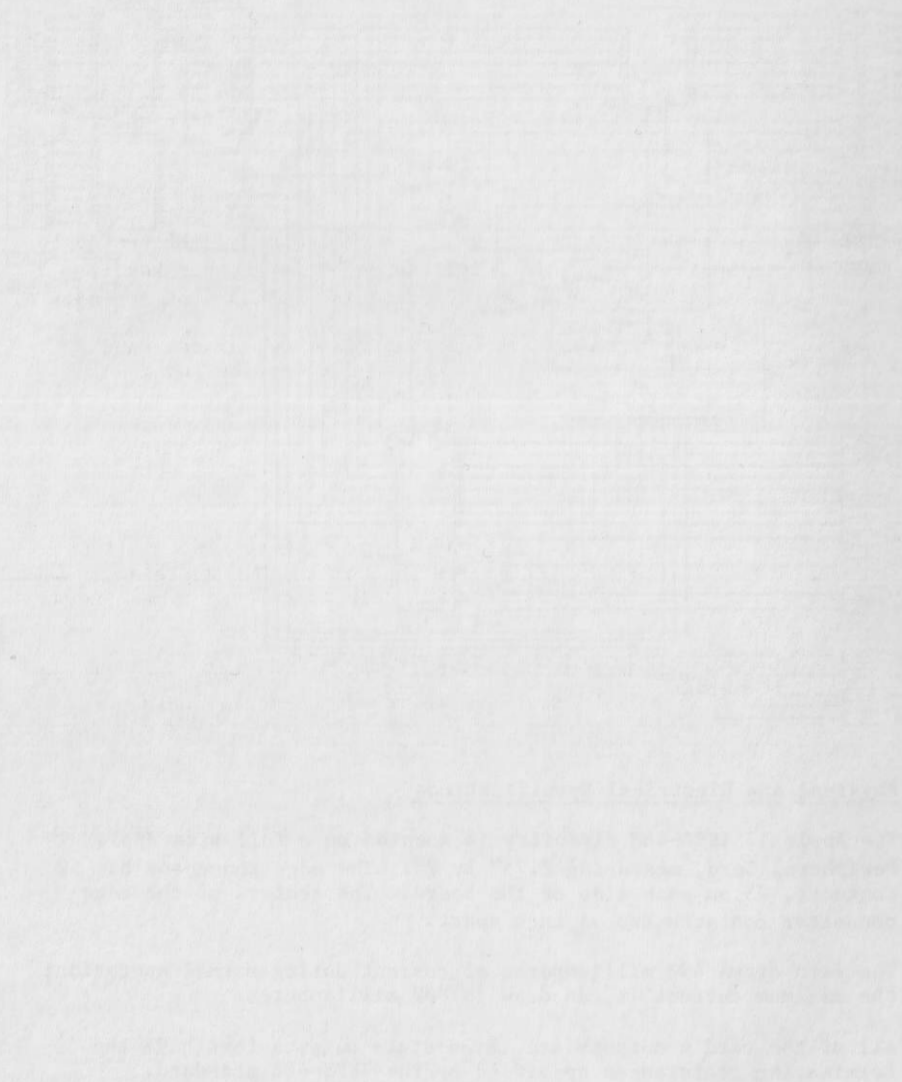
### Physical and Electrical Specifications

The Apple II IEEE-488 circuitry is mounted on a full size Apple Peripheral Card, measuring 2.75" by 7". The edge connector has 50 contacts, 25 on each side of the board. The centers of the edge connector contacts are .1 inch apart.

The card draws 400 milliamperes of current during normal operation; the maximum current it can draw is 800 milliamperes.

All of the card's outputs are three-state outputs that have the terminating resistances specified by the IEEE-488 standard.

# SCHEMATIC DIAGRAM



# GLOSSARY

- APPLESOFT:** The floating-point version of BASIC that your Apple uses. It has the CHR\$ function that must be used to generate secondary addresses.
- Asynchronous:** Not synchronized. Synchronous events are generally clocked by the same clock pulse, asynchronous events are not.
- Buffered Input:** Data that is sent to a device faster than the device can process it can be stored temporarily in a special input storage area (an input buffer) and then processed by the device whenever the device is ready. Input buffers are generally internal to devices.
- Control Function:** A function that tells some device what to do.
- CTRL:** The label that appears on a key on the left side of the Apple II's keyboard. It should be held down while another key is pressed to generate the control codes in Table A-4 of Appendix A.
- Controller:** The device that controls all transfers of data and control bytes on the bus. The controller for the bus is the IEEE-488 card together with the Apple.
- Daisy Chained:** Devices that are attached to a bus one after the other are said to be daisy chained. Devices can also be star connected on the bus.
- Direct Memory Access (DMA):** High speed accessing of memory without the use of the CPU (in this case the 6502). The Apple's processor can be suspended for DMA using pin 22 of a peripheral I/O connector slot.
- Handshake:** A sequence of signals transmitted over a bus that allows devices to identify themselves as ready to receive data, transmit data, or neither. The three handshake lines for any IEEE-488 bus are NRFD, DAV and NDAC.
- Integer BASIC:** The Apple version of BASIC that does not have the use of floating-point numbers. The CHR\$ function may be simulated by using Program 1 from Appendix D.
- Interface:** As used in this manual, the hardware and software that allows two devices to communicate.
- Jumper:** One or more wires (sometimes made into a plug) connected in one of several alternative ways. Just as you can use jumpers to set the memory configuration of the Apple, manufacturers often use jumpers to set the address to which a device responds.

**Listener:** A device that is enabled to receive data over a bus.

**Local Operation:** Operation of a device using the knobs and switches on the device front panel, as opposed to remote operation on the GPIB.

**Parallel Poll:** A method of simultaneously asking many (for GPIB, up to eight) devices if they are waiting to be serviced. A device that is waiting to be serviced will respond to the poll by setting a bit in the status byte read by the PP command.

**Piggyback Connectors:** Connectors that have plugs above their pins, allowing more than one cable to be connected at the same point.

**Remote Operation:** Operation of a device using control information device (for example, the controller on the GPIB).

**Serial Poll:** A method of polling that asks each device in turn if it needs servicing. This method is slower than parallel polling but it can be used with more devices. A serial poll is initiated after the SRQ line is pulled low by a device, and it uses the SRQ command.

**Standard:** A way of doing something that has been defined by a standards committee. Products that adhere to a single standard are compatible with each other. For example, the Apple GPIB adheres to the IEEE-488 standard.

**Star Connected:** A bus configuration in which one device is like the hub of a wheel with the other devices on the rim of the wheel, connected to the first device along the spokes. This configuration is not recommended.

**Talker:** A device that is enabled to transmit data over a bus. The IEEE-488 standard permits only one talker at a time.

**Term:** Short for terminator.

**Terminator:** A particular symbol or character that is used to mark the end of something (for example, a string of data). <CR>, <EOS> and EOI are all used as GPIB terminators.

**TMS 9914** - The GPIB adaptor chip around which the Apple GPIB is designed. It is made by Texas Instruments.

**Trigger:** To cause something to happen immediately. Triggers are used for synchronization or enabling events.

# INDEX

## A

AB command 22, 37, 46, 69  
ABORT: see AB command  
acceptor handshake function 17  
active low logic 14  
address  
    my listen 43  
    my talk 43  
addresses 24  
    conventions 2  
    listener 24, 39  
    secondary 23-24, 41  
    talker 24, 40  
addressing switches 25  
AHL function 17  
Apple IEEE interconnection cable 4  
Apple II IEEE-488 72  
Apple Language Card 4  
Apple Pascal Operating System 65  
Apple 6502 Assembler/Editor 55, 63, 65  
APPLESOFT 73  
ASCII codes 39-42  
Assembler/Editor, Apple 6502 55, 63, 65  
asynchronous 73  
ATN line 13-14  
ATteNtion line 14

## B

Backus-Naur Form: see BNF  
basic listener function 17  
basic talker function 17  
block diagram 9  
block transfers 55  
BNF 20-23  
buffered input 12, 73  
buffers, data 61  
bus  
    IEEE-488 1  
    maximum length 10  
    transfers 27  
BUS error 38, 52  
BUS? 20, 22

## C

CA command 22, 32, 45, 69  
cables  
    Apple IEEE interconnection 4  
    IEEE standard 12-16  
card connector 5  
CHR\$ function 24, 29, 40-42  
    for Integer BASIC 51  
circuitry 72  
CL command 22, 33, 45  
clamp, metal connector 5  
clear all interfaces 37  
clear device 43  
clear select device 43  
CLEAR: see CL command  
closing the Apple 7  
CLRAL: see CA command  
CMD error 38  
CNT error 38  
commands 20-37  
    data transfer 27  
    descriptions 25  
    protocol 26  
    set operating characteristics 27  
    string 20-21, 27  
commas, input with 54  
configure parallel poll 43  
connections, GPIB 13  
connector slot 1, 4  
connectors  
    card 5  
    IEEE 5  
    piggyback 9  
control characters parameter 20  
control codes 42  
control function 73  
control lines 9, 13, 14  
controller 9, 73  
controller function 18  
count parameter 21-26, 30-31  
CTRL 73  
current 72  
C1 function 18  
C2 function 18  
C3 function 17  
C4 function 17  
C25 function 17

## D

- daisy chained 9-11, 73
- data buffers 61
- data lines 9-14
- data parameter 20-23
- data rates 12
  - fast transfer 62
- data string 20-23, 54
- Data Valid line 16
- DAV line 13, 16
- DC function 17
- DCL 43
- decimal digit 23
- default values 19
  - don't reset 26
  - reset 26
- device clear 43
- device clear function 17
- device number 23
- device operating features 24
- device trigger function 18
- devices
  - maximum number 9-10
  - test 2
- DEVICE: see DV command
- direct memory access 12, 73
- disable parallel poll 37, 43
- disable serial poll 43
- disk controller card 4
- DMA 12, 73
- DOS
  - commands 26
  - dimension statements 53
  - I/O locations 53
- DT function 18
- DV command 19, 22, 27, 29, 47, 69

## E

- edge connector 72
- electrical requirements 13
- enable 22-23
- enable byte 36
- enable list 23, 36
- enable parallel poll 36, 43
- enable serial poll 43
- End Or Identify: see EOI line
- EOI line 13-14, 29-31, 58-59
- EOS: see ES command

- EOS character 23, 27, 30-31, 58-59
- error messages 38
- ES command 22, 27-28, 47

## F

- fast transfers routines 55-63
- functions
  - control 73
  - IEEE 16-18

## G

- GET statement 43, 54
- go to local 43
- GPIO 1
  - connections 13
  - error messages 38
  - listener addresses 39
  - secondary addresses 41
  - talker addresses 40
- ground 6
  - shield 5, 13
- group execute trigger 18, 32, 43
- GTL 43

## H

- handshake 14, 73
- hex digit 23
- HP 3438A Multimeter 67
- HP 3586C Selective Level Meter 67-68
- HP-IB 1

## I

- IEEE connector 5
- IEEE Standard 488 1-2
- IEEE Standard screw 3, 6
- IEEE-488 bus 1
- IEEE-488 card 1, 72
- IFC line 13, 15
- input, buffered 73
- INPUT statement 25, 54



input with commas 54  
installation 3-7  
instrument modes, reset 20  
Integer BASIC 73  
interface 73  
interface cards  
    Apple Language 4  
    disk controller 4  
    IEEE-488 1  
interface clear function 18  
InterFace Clear line 15  
IN#n 19, 26, 52-53  
I/O registers 55

## J

jumper 25, 73

## K

## L

LA command 22, 33-34, 45  
length, bus 10  
LF command 19, 22, 27, 29-31,  
    46, 67, 69  
LINEFEED: see LF command  
lines  
    ATN 14  
    DAV 16  
    EOI 14  
    IFC 15  
    NDAC 15  
    NRFD 15  
    REN 15  
    SRQ 15  
listen list 21-26, 29-34, 37  
listener 9, 23, 74  
    primary 23  
listener addresses 24, 39  
LL command 22, 34, 45, 69  
LLKAL: see LL command  
LLO 43  
LO command 22, 34, 45  
local lockout 43  
local operation 74  
LOCAL: see LA command  
LOCDEV: see LO command

LST error 38  
LSTN bus sequence 43  
L1 function 17

## M

maximums  
    bus length 10  
    number of devices 9  
    current 72  
metal connector clamp 5  
metallic interior 6  
MLA 43  
MTA 43  
my listen address 43  
my talk address 43

## N

NDAC line 13, 15  
negative logic 14  
Not Data ACcepted line 15  
Not Ready For Data line 15  
NRFD line 13, 15  
number, device 23  
number of devices, maximum 9-10

## O

off/on 23  
ONERR GOTO statement 38  
ONERR routine 52  
opening the Apple 3  
operation  
    local 74  
    remote 74  
output data format 19

## P

parallel poll 36, 74  
    configure 43  
    disable 37, 43  
    enable 36, 43  
    function 17  
    unconfigure 37, 43

Pascal 55, 65  
 PD command 22, 37, 46  
 PE command 22, 36, 46  
 peripheral connector slot 1, 4  
 piggyback connectors 9, 74  
 PP command 22, 36, 46  
 PPC 43  
 PPD 43  
 PPDIS: see PD command  
 PPE 43  
 PPENB: see PE command  
 PPOLL: see PP command  
 PPU 43  
 PPUAL: see PU command  
 primary listener 23  
 primary talker 23  
 PRINT statement 25, 27, 54  
 programming the GPIB 19  
 programs, sample 67  
 PR#n 19, 26, 28, 29, 52-53  
 PU command 22, 46

## Q

## R

RA command 22, 33, 45, 69  
 RC command 22, 31, 44  
 RD command 22, 30, 44, 53, 55, 67-68  
 read only memory 1  
 READCNT: see RC command  
 READ: see RD command  
 receive routine 59  
 registers  
   I/O 55  
   9914 49  
 REMAL: see RA command  
 REMDV: see RM command  
 remote enable function 17  
 Remote ENable line 15  
 remote operation 74  
 REN line 13, 15, 33-34  
 requirements, electrical 13  
 reset card default values 19, 26  
 reset instruction 28  
 reset instrument modes 20  
 RESET key 29, 34  
 respond to service request  
   function 17

RM command 22, 33, 45  
 ROM: see read only memory 1

## S

sample programs 67  
 SC command 19, 22, 26, 27-28, 47, 52-53, 55, 67, 69  
 SCREEN: see SC command  
 screw 3, 6, 7  
 SDC 43  
 secondary address 23-24, 41  
 select device clear 43  
 semicolon 27, 31  
 send routine 58  
 serial poll 35, 74  
   disable 43  
   enable 43  
 Service ReQuest line 15  
 shield ground 5, 13  
 SHI function 16  
 skip reinitialize function 53  
 slot, peripheral connector 1, 4  
 source handshake function 16  
 SP command 22, 35, 45  
 SPD 43  
 SPE 43  
 special symbols 2  
 specifications 72  
 SPOLL: see SP command  
 SR command 22, 35, 45, 53  
 srq 23  
 SRQ line 13, 15, 35  
 SRQD: see SR command  
 standard 74  
 star connected 9-11, 74  
 static discharge 3  
 status word 23, 35-36  
 switches, addressing 25

## T

TALK bus sequence 43  
 talker 9, 22-24, 30-32, 35, 74  
   addresses 24, 40  
   primary 23  
 term 22-23, 30-31, 74  
 terminator: see term  
 test device 2  
 TG command 20, 22, 32, 44, 67-68

three-state operation 72  
three-wire handshake 15  
TMS 9914 49, 55, 57, 65, 74  
transfer control lines 15  
trigger 74  
TRIGR: see TG command  
T3 function 17

## U

unconfigure parallel poll 37, 43  
universal unlisten command 24  
universal untalk command 24  
UNL command 24, 43  
unlisten: see UNL command  
UNT command 43  
untalk: see UNT command  
UNTALK: see UT command  
using addresses 25  
UT command 22, 37, 47

## V

## W

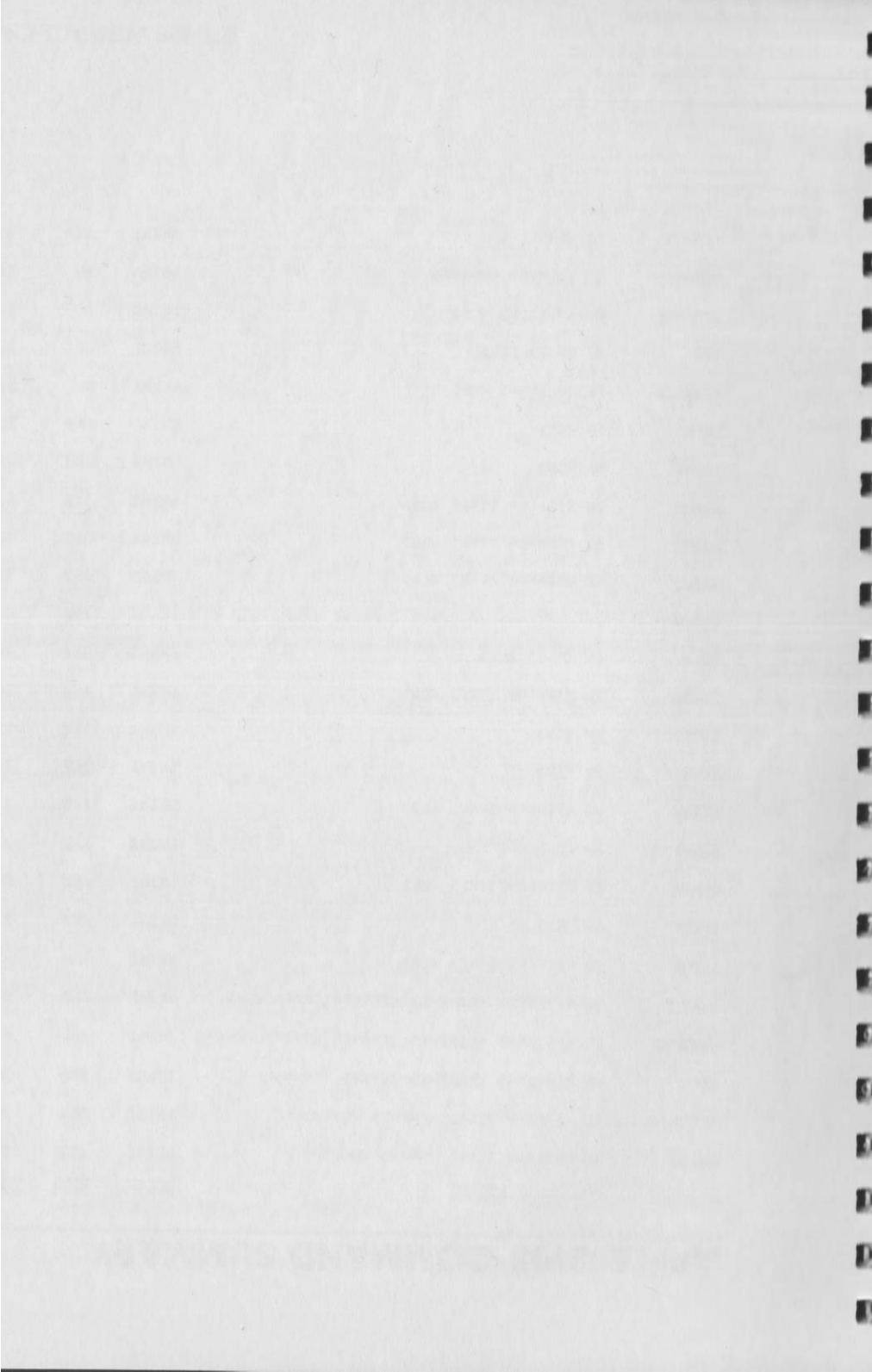
wait routine 60  
WC command 22, 30, 44, 53  
WRITECNT: see WC command  
WRITE: see WT command  
WT command 22, 29, 44, 46-47,  
52-53, 55, 68

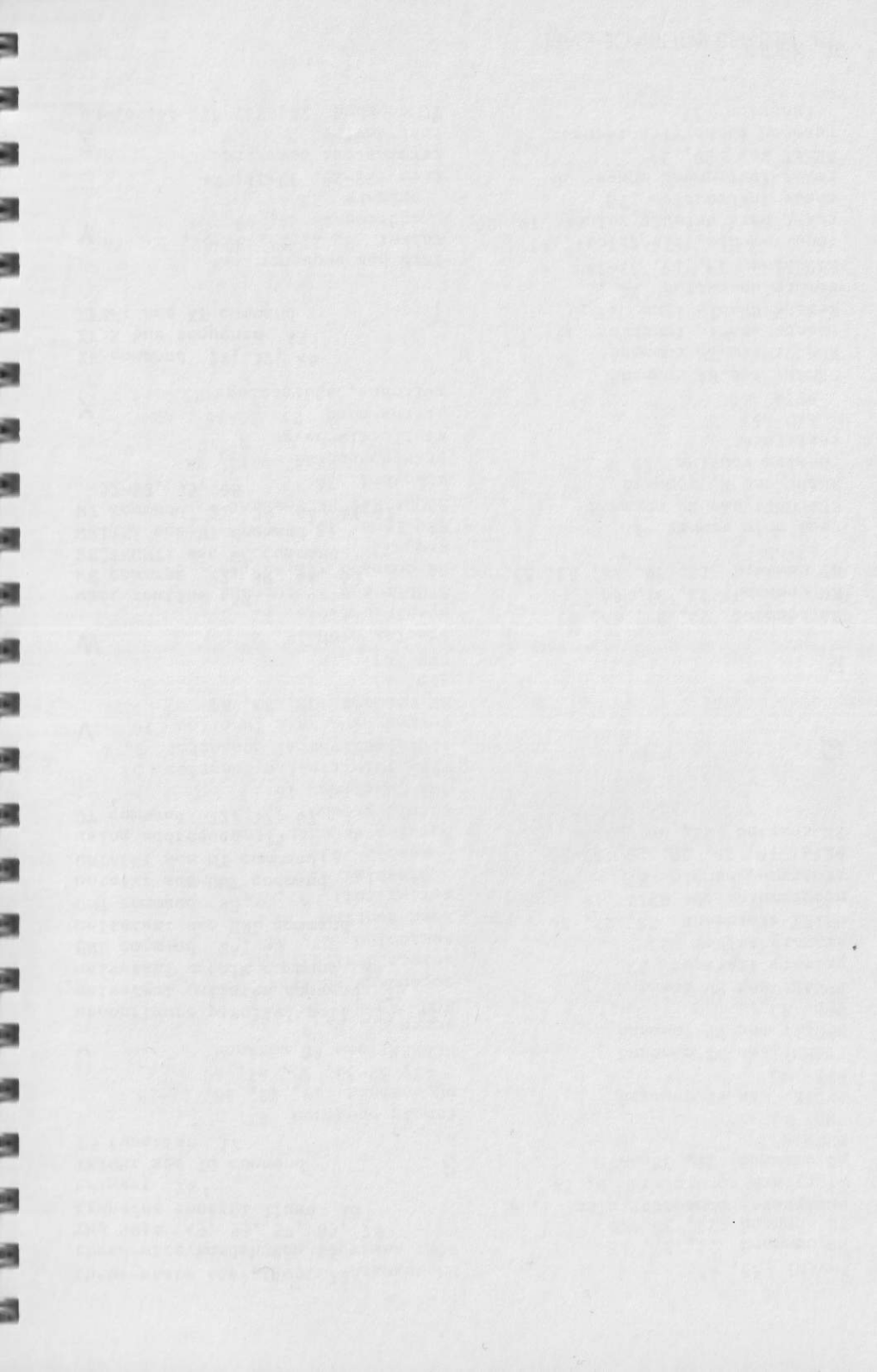
## X

XF command 22, 32, 44  
XFER bus sequence 43  
XFER: see XF command

## Y

## Z







# APPLE GPIB COMMAND SUMMARY

<u>COMMAND</u>	<u>FORMAT</u>	<u>BASIC</u>	<u>BUS?</u>	<u>PAGE</u>
WRITE	WT <listen list> <data> <EOS>	PRINT	YES	29
WRITECNT	WC <listen list> <count> <data>	PRINT	YES	30
READ	RD <talker> <CTRL-Z> <data> <term>	INPUT	YES	30
READCNT	RC <talker> <CTRL-Z> <count> <data> <term>	INPUT	YES	31
XFER	XF <talker> <CTRL-Z> <listen list> <CR>	PRINT	YES	32
TRIGR	TG <listen list> <CR>	PRINT	YES	32
CLRAL	CA <CR>	PRINT	YES	32
CLEAR	CL <listen list> <CR>	PRINT	YES	33
REMAI	RA <CR>	PRINT	YES	33
REMDV	RM <listen list> <CR>	PRINT	YES	33
LLKAL	LL <CR>	PRINT	YES	34
LOCAL	LA <CR>	PRINT	YES	34
LOCDV	LO <listen list> <CR>	PRINT	YES	34
SRQD	SR <srq> <CR>	INPUT	YES	35
SPOLL	SP <talker> <CTRL-Z> <status word> <CR>	INPUT	YES	35
PPOLL	PP <status word> <CR>	INPUT	YES	36
PPENB	PE <enable list> <CR>	PRINT	YES	36
PPDIS	PD <listen list> <CR>	PRINT	YES	37
PPUAL	PU <CR>	PRINT	YES	37
ABORT	AB <CR>	PRINT	YES	37
LINEFEED	LF <off/on> <CR>	PRINT	NO	27
EOS	ES <EOS> <CR>	PRINT	NO	28
SCREEN	SC <off/on> <CR>	PRINT	NO	28
DEVICE	DV <device number>	PRINT	NO	29
UNTALK	UT <CR>	PRINT	YES	37

# APPLE GPIB COMMAND STRINGS

---

<listen list> ::= <CTRL-Z> | <listener> <CTRL-Z> | <listener> <listen list>

<listener> ::= <primary listener> | <primary listener> <secondary address>

\* <primary listener> ::= <SP> | ! | " | # ... > | ?

\* <secondary address> ::= ' | a | b | c ... } | ~

<count> ::= <decimal digit> <CTRL-Z> | <decimal digit> <count>  
NOTE: The maximum total value for count is 255.

<EOS> ::= any ASCII character

<data> ::= empty | any ASCII character | any ASCII character <data>

<talker> ::= empty | <primary talker> | <primary talker> <secondary address>

\* <primary talker> ::= @ | A | B | C ... ^ | \_

\* <hex digit> ::= 0 | 1 | 2 ... 9 | A | B ... F

\* <decimal digit> ::= 0 | 1 | 2 ... 9

<status word> ::= <hex digit> <hex digit>

<srq> ::= ASCII "T" | ASCII "F"

<term> ::= empty | <CR> | <EOS> | EOI  
NOTE: empty if last char = <CR> or <CR><LF>

<device number> ::= <decimal digit> <decimal digit>  
NOTE: Maximum device number is 30

<off/on> ::= ASCII "0" | ASCII "1"

<enable list> ::= <listener> <enable> | <listener> <enable> <enable list>

\* <enable> ::= @ | A | B ... N | O

<CR> ::= ASCII carriage return (13 decimal)

<LF> ::= ASCII linefeed (10 decimal)

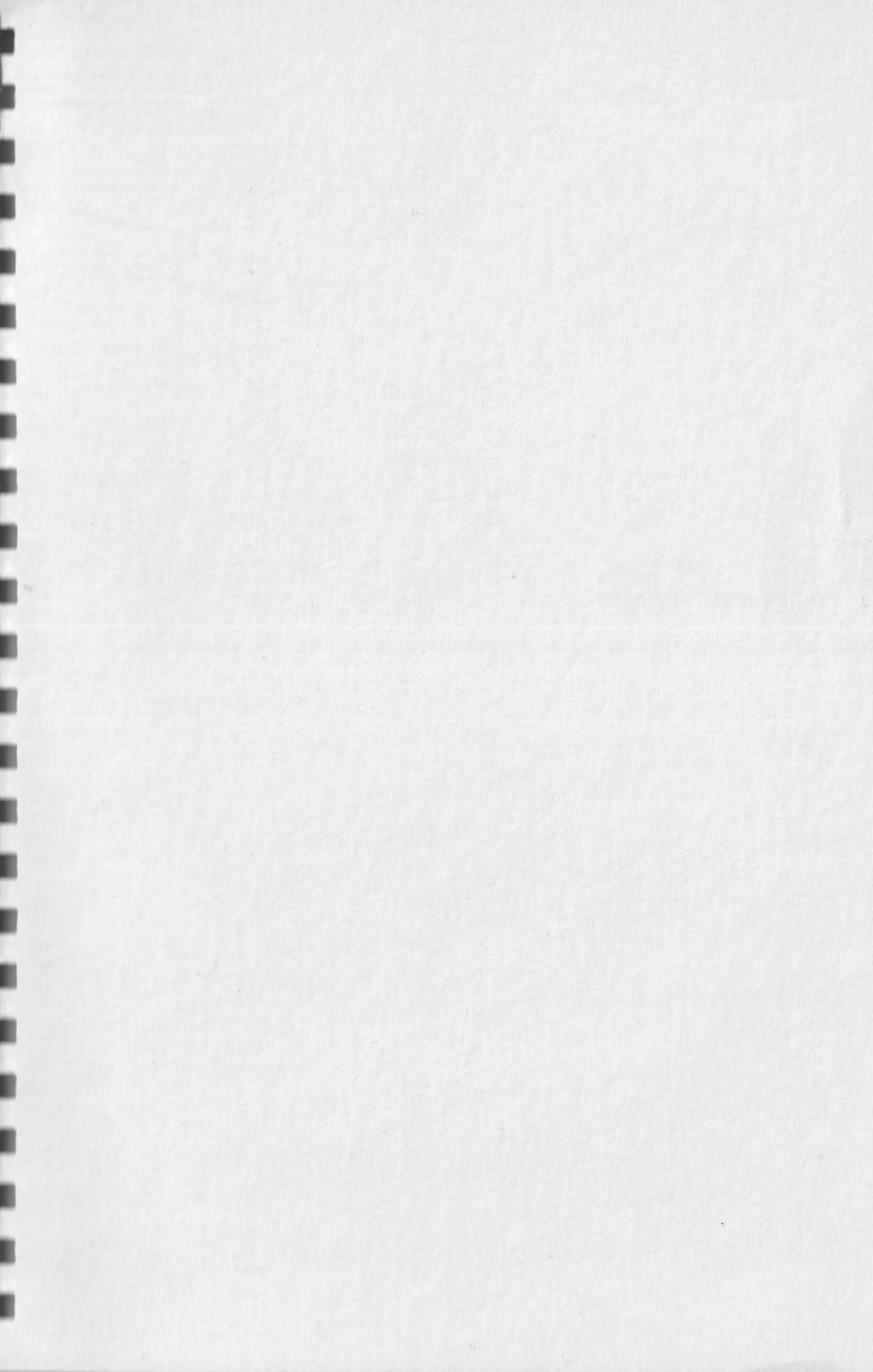
<SP> ::= ASCII space (32 decimal)

<CTRL-Z> ::= ASCII SUB (26 decimal)

<CTRL-D> ::= ASCII ET (4 decimal)

\* NOTE: Marked elements are ASCII characters.







10260 Bandley Drive  
Cupertino, California 95014  
(408) 996-1010